UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

Estudi i implementació d'una tasca robotitzada capaç de replicar en 3D una figura reconeguda mitjançant visió artificial

Autor:

Ivan Chueco Martínez

Dirigit per:

Rita Mª Planas Dangla

Grau en Enginyeria Electrònica Industrial i Automàtica

Convocatòria Gener 2018

Índex

Índex d'il·lustracions

Resum

En aquest document es descriuen els aspectes més importants de les aplicacions desenvolupades per dur a terme una tasca robotitzada mitjançant visió artificial. L'objectiu principal d'aquest treball tracta de replicar una figura creada amb envasos de diferents formes i colors dipositada sobre una taula, que es troba en un entorn de treball petit al voltant de l'abast del robot i dins de l'espai de visió de la càmera que inicialment recull les dades.

Per l'obtenció d'informació es fa servir una càmera web instal·lada al sostre del laboratori, propera a la vertical de l'àrea frontal de treball del robot.

El conjunt dissenyat consta d'un primer programa que activa la comunicació amb la càmera per l'obtenció de la visió, fa un tractament d'imatge per identificar els objectes del nostre interès i realitza els càlculs necessaris per obtenir el posicionament dels objectes que formen la figura a replicar.

El segon programa dissenyat fa servir les dades obtingudes per moure el robot, de forma que agafa els envasos utilitzats en la figura presentada anteriorment i els diposita en la mateixa posició.

Paraules clau

Visió artificial, tractament d'imatge, robot, posicionament, replicar

1. Objectius

L'objectiu principal del projecte és dissenyar una tasca robotitzada capaç de replicar una figura creada amb diferents elements de tres dimensions fent ús de visió artificial mitjançant una càmera web i d'un robot industrial de sis eixos. Els requisits imprescindibles per controlar el sistema són:

- Crear un sistema de reconeixement d'imatge flexible que no requereixi precisió en l'ús de l'àrea de treball, ja que cal poder muntar i treure la taula de treball sovint a causa dels diversos usos del robot.
- Dissenyar un sistema de reconeixement de formes i colors amb un rang prou ampli per acceptar diferents condicions de llum del laboratori.
- Obtenir la informació de la posició i orientació de les peces que componen la figura en coordenades del robot a partir de la imatge.
- Facilitar la comunicació entre els dos llenguatges de programació emprats.

2. Plantejament del problema

El sistema que es vol desenvolupar tracta de reproduir un sistema de paletització controlat únicament per visió artificial a partir d'una càmera. Partint del material que trobem al laboratori com són un robot industrial i una càmera web volem poder presentar una figura sobre una taula davant de l'esmentat robot, realitzar una fotografia de la distribució de diversos objectes formant una figura concreta, processar aquesta imatge de forma que obtinguem les dades necessàries per saber la ubicació i posició de cada objecte que forma part de la figura, traspassar aquestes dades a la controladora del robot, deixar els objectes fets servir en un emmagatzematge de posició fixa a l'abast del robot i que aquest reprodueixi la mateixa figura que havíem presentat inicialment.

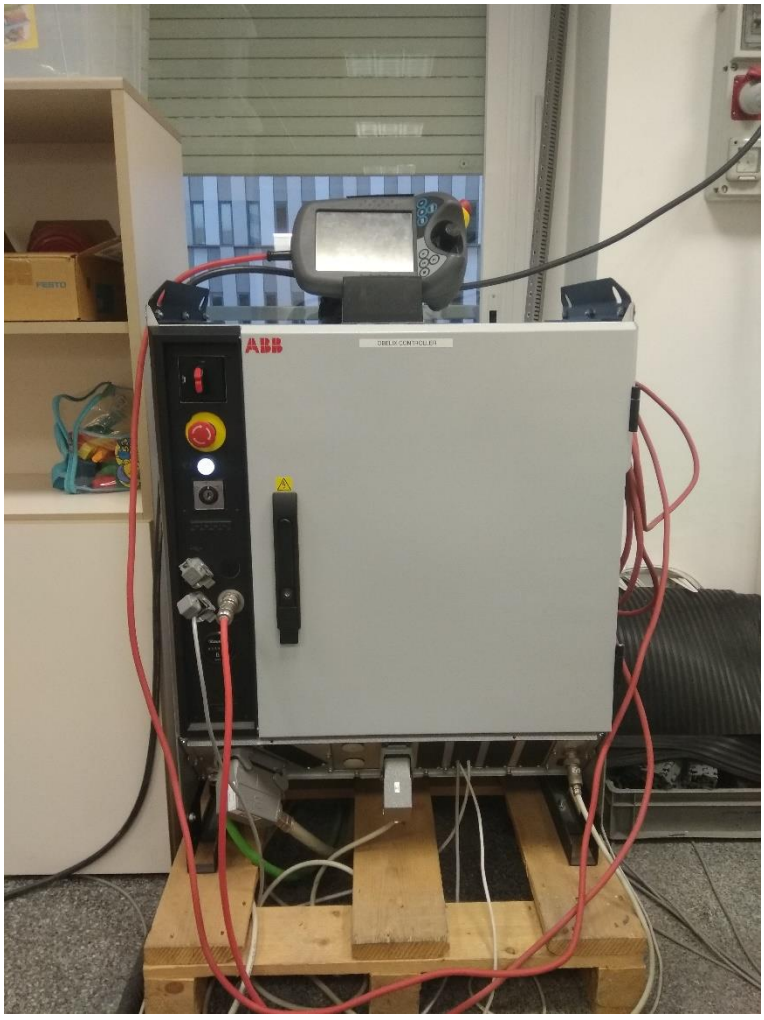
2.1. Abast

Com que el contingut del projecte requereix diversos materials i eines que ja podíem trobar al laboratori o que necessàriament havíem de fer servir i d'altres que no, presentarem un resum distingint ambdós grups.

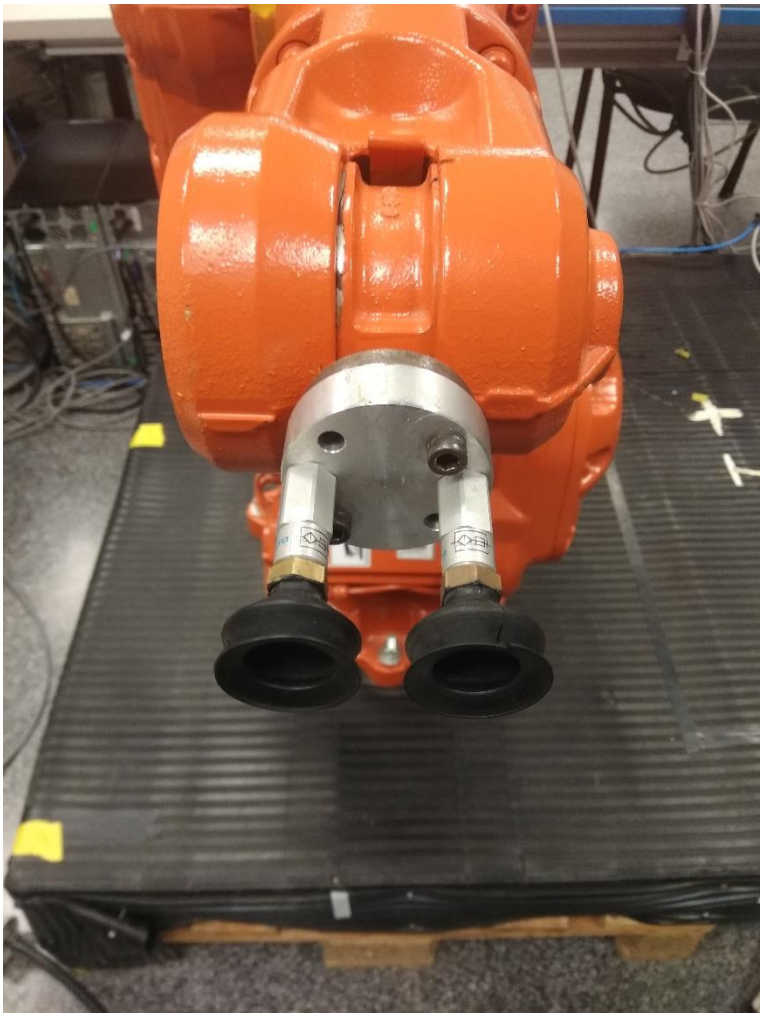
Els materials i eines que ens han vingut donats per l'entorn de treball han estat els següents:

- Robot i controladora
  Es tracta d'un braç robòtic model IRB 140 i una controladora model IRC5 amb una *Teach Pendant* (Terminal de programació tàctil i portàtil), ambdós de la companyia ABB.
  Això ens condiciona, per tant, a treballar amb un robot hidràulic i amb l'eina que porta instal·lada, un motor de succió d'aire comprimit amb una eina incorporada en el sisè eix del robot que consta de dues ventoses.

Il·lustració 1 Robot Industrial IRB 140 ABB



Il·lustració 2 Controladora del robot



Il·lustració 3 Detall ventoses (eina de treball del robot)

- Càmera web
  Model M24M Allround L22 de la marca MOBOTIX amb connexió IP via cable, i el software de calibratge i manipulació *MxCC* o *MOBOTIX Control Center* (centre de control de MOBOTIX) d'aquesta.

Il·lustració 4 Càmera web

- Ordinador connectat a la controladora del robot.
  Aquest permet accés als directoris de fitxers de la controladora i facilita la introducció de fitxers amb què treballa aquesta. L'ordinador treballa amb un sistema operatiu Windows 7 i treballa sense possibilitat de connexió a internet.

- Llenguatge de programació RAPID
  És el llenguatge que la controladora és capaç d'interpretar, fet pel qual s'han d'escriure les tasques i moviments que ha de fer el robot en aquest codi de programació. Es tracta d'un llenguatge d'alt nivell amb molta similitud al llenguatge de programació C++, enfocat a la manipulació d'objectes.

Els materials i eines que en canvi hem pogut escollir han estat:

- Objectes per formar les figures
  Es tracta de 7 models diferents d'envasos per guardar menjar. Aquests tenen diferents alçades, formes (un quadrat, un rodó i 5 rectangulars), diferents mides i tapes amb 5 colors diferents (rosa, vermell, taronja, blau i verd). Aquests envasos s'han aprofitat, ja que es trobaven al laboratori com a material d'altres projectes, i satisfan la necessitat d'haver d'identificar diferents objectes en la figura a reproduir.

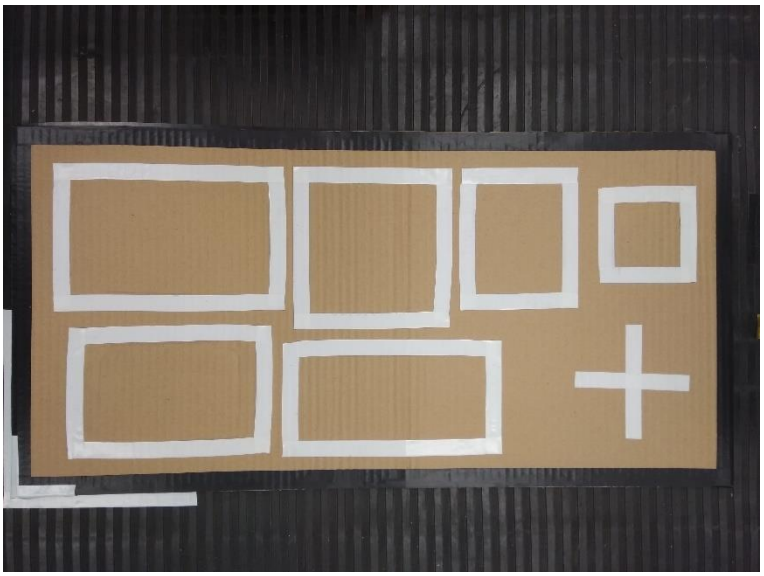
Il·lustració 5 Envasos de colors

- Taula de treball

  Com en el cas anterior, al laboratori es troba una taula de les dimensions necessàries que va pertànyer a un altre projecte, i que s'ha escollit per presentar sota la càmera i davant del robot la figura a reproduir.

Il·lustració 6 Taula (espai de treball)

- Magatzem d'objectes
  Per donar al robot una ubicació concreta d'on agafar els envasos que ha de col·locar sobre la taula s'ha creat una plataforma de cartó amb marques per situar aquests envasos sempre en la mateixa posició, i s'ha marcat la ubicació d'aquesta sobre l'entarimat on es troba el robot.



Il·lustració 7 Magatzem d'envasos

- Llenguatge de programació C# (C-Sharp) en entorn *Visual Studio* 2015
  Entre dues opcions inicials plantejades per la directora del projecte com eren C# i Visual Basic, s'ha escollit C#.

Amb aquesta informació, el plantejament del problema a treballar tracta de les següents parts:

1. Presentar sobre una taula una figura formada amb diversos envasos de colors.
2. Obtenir amb la càmera web MOBOTIX instal·lada fixa al sostre una imatge de la distribució, amb programació en C#.
3. Processar la imatge per identificar els diversos tipus d'envàs i la posició i orientació d'aquests, i guardar les dades en una taula de dades, amb programació en C#.
4. Transmetre la informació obtinguda al llenguatge de programació RAPID.
5. Dur a terme la tasca robotitzada de replicació de la imatge en llenguatge de programació RAPID.

És per això que el projecte inclou els següents punts:

- Tractament i manipulació d'imatge
- Càlculs espacials
- Transformació matemàtica de dades
- Control de trajectòria
- Control espacial

3. Antecedents i estat de l'art

El sistema que es crea en aquest projecte no és una innovació, ja que actualment és un dels elements que més s'investiga i utilitza a la indústria en diversos aspectes. Podem trobar sovint l'ús de robots amb visió artificial en tasques de selecció d'elements en cintes transportadores, revisió i eliminació de productes amb desperfectes en un procés de fabricació com també en tasques de paletització o apilament, entre d'altres.

Podem trobar grans marques, com per exemple *Festo*, en què presenten els conjunts de visió artificial i posicionament servo-pneumàtic en l'apartat "Innovació" de la seva pàgina de presentació, o altres companyies com *Bosch* o *Vision & Control* que ofereixen aquests sistemes com a productes preparats per a empreses de fabricació i de control de qualitat.

Com que el projecte dut a terme no tracta de crear un projecte innovador, la tasca duta a terme vol extreure un gran aprenentatge personal aconseguint unir diversos dispositius del laboratori amb l'objectiu de fer un sistema funcional unitari que desenvolupi la tasca plantejada, de la mateixa manera que es faria en una empresa que vol reutilitzar material per fer una nova estació de treball.

4. Plantejament i justificació de solucions

En el marc de treball presentat han calgut diverses decisions davant de les possibles solucions que es podien dur a terme per resoldre els objectius plantejats. En aquest apartat es descriuen aquestes solucions per finalment explicar quina ha estat l'opció escollida i el motiu.

4.1. Llenguatges de programació

Com ha estat esmentat anteriorment, la proposta inicial de la directora del projecte demanava escollir entre els dos llenguatges de programació més utilitzats de l'entorn *Visual Studio* de Microsoft, Visual Basic .NET i C# (C-Sharp).

Fent una recerca de les diferències entre els dos llenguatges trobem que Microsoft els considera igual de poderosos, així com usuaris amb alt nivell de coneixements de programació. Ambdós llenguatges es compilen a un mateix llenguatge entremig, que converteix les instruccions a codi màquina amb un compilador abans de ser executat. La comparació entre dos programes simples com el programa d'exemple Hola món demostra que el codi màquina d'ambdós és pràcticament el mateix. És per això que la gran diferència entre aquests és purament de sintaxi.

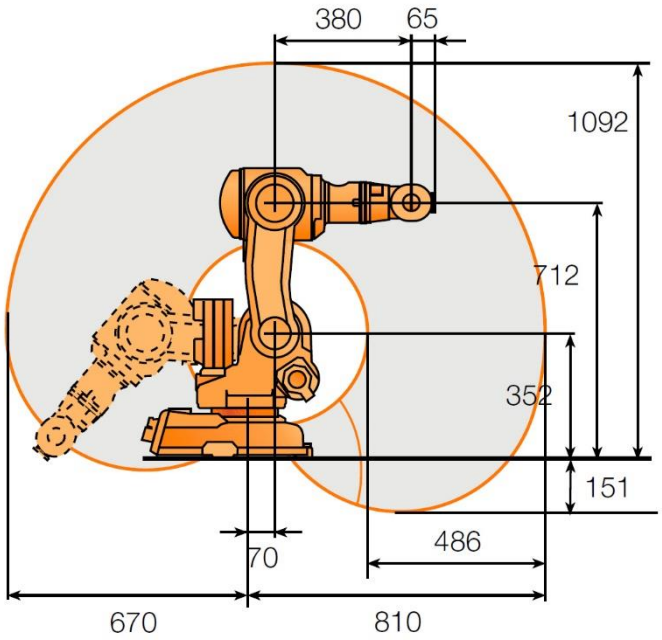
Per tant, amb Visual Basic podem observar l'ús de majúscules en la primera lletra de la definició de tipus de variables, mots reservats o marcadors d'inici de nova línia; la utilització de paraules per relacionar conceptes entre sí (And, Or, Not, AndAlso...) o el tancament de blocs fent ús de la paraula End seguida del bloc tancat (End Sub, End For, End While).

Per altra banda, usant els mateixos exemples, amb C# observem que s'empren les majúscules només en la declaració de variables concretes i en algunes paraules reservades; en la relació entre conceptes es fa ús majoritàriament de símbols (&, |, !. &&...) i la definició de blocs va marcada per claudàtors i gafets ({}, []).

La solució escollida ha estat utilitzar C#. Això és perquè el resultat d'ambdós llenguatges proporciona la mateixa qualitat final però la sintaxi de C# té més similitud amb els llenguatges cursats en assignatures de la carrera que també es basen en llenguatge de programació C orientat a objectes, i això em facilita l'aprenentatge i el desenvolupament en l'entorn de programació permetent-me treballar més eficientment.

4.2. Espai de treball del robot

L'àrea de treball del robot és limitada i ve definida pel model de robot, ja que depèn de la suma de posicions dels eixos d'aquest. En analitzar cadascuna de les posicions diferents en que podem trobar el robot i presentar-les en un sol dibuix, podem definir l'àrea de treball del robot. En el nostre cas, el robot IRB 140 d'ABB té la capacitat de treballar a una distància màxima respecte del seu centre de coordenades XY de 810 mm, en una semi corona circular d'aproximadament 270 graus al voltant del segon eix, i en una àrea de revolució de 360 graus, com es pot observar en la figura següent:

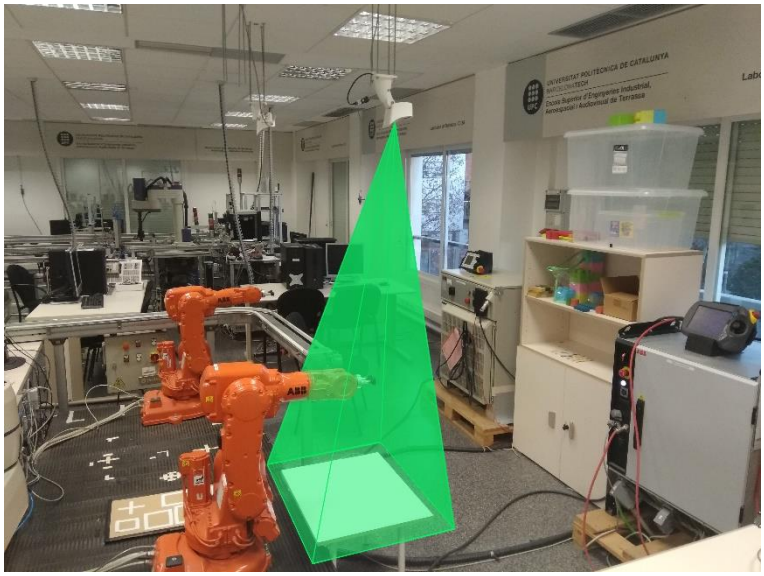


Il·lustració 8 Àrea de treball planar IRB 140 d'ABB

Aquest robot es troba situat sobre una plataforma que situa l'alçada de la base a 290 mm respecte del terra del laboratori.

Partint de les limitacions físiques de l'àrea de treball que tenim, cal definir quins espais volem destinar a cada tasca.

Una restricció inicial que trobem és la posició fixa de la càmera web, que és la visió artificial de la qual hem d'obtenir la figura que voldrem replicar. Aquesta càmera està situada a 2050 mm respecte del terra del laboratori, i també aproximadament a 800 mm de distància frontal del centre del robot i desplaçat 500 mm a l'esquerra del mateix centre:

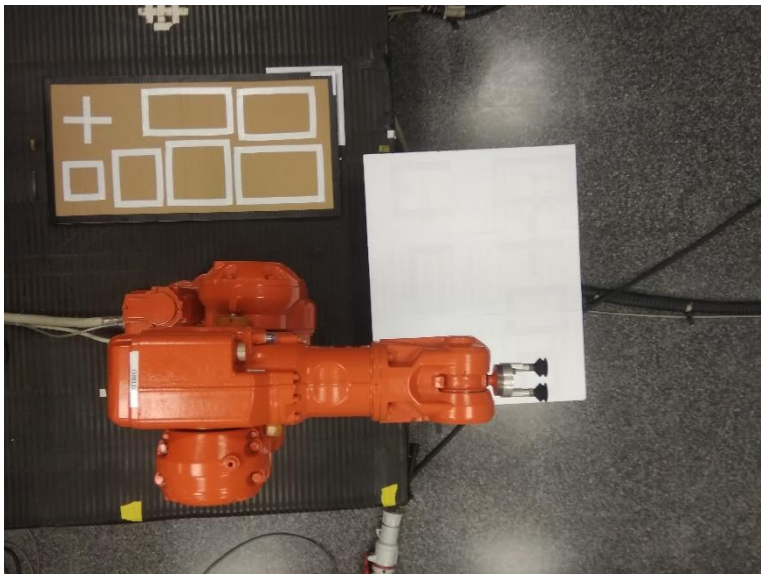


Il·lustració 9 Espai de visió de la càmera

Com la càmera ens limita a fer la figura reproduïble a l'espai de visió d'aquesta, definim que és la zona frontal on es situarà l'inici de l'exercici. Per facilitar la maniobrabilitat en la manipulació dels envasos i salvar l'alçada de la plataforma sobre la qual se situa el robot farem servir una taula auxiliar d'una alçada de 420 mm i unes dimensions de 600 mm per 500 mm disposada horitzontalment. Aquesta taula s'aprofita d'un antic projectista que la va cedir al laboratori.

L'altre espai que ens cal definir és el dipòsit d'objectes al que caldrà accedir el robot per fer la rèplica. Aquest espai ens interessa que tingui una ubicació fixa, ja que no podem controlar-la per visió artificial a causa de la falta d'espai, i també perquè ens facilita la programació dels moviments del robot al reduir la quantitat de dades variables. Partint de l'àrea de treball del robot, la situació escollida és l'espai lateral esquerre a continuació del robot, sobre la plataforma on aquest se situa. Es fabrica una superfície de dimensions 700 mm per 350 mm amb indicadors de la posició de cada envàs diferent i es situa una marca sobre la plataforma per facilitar la col·locació d'aquesta.

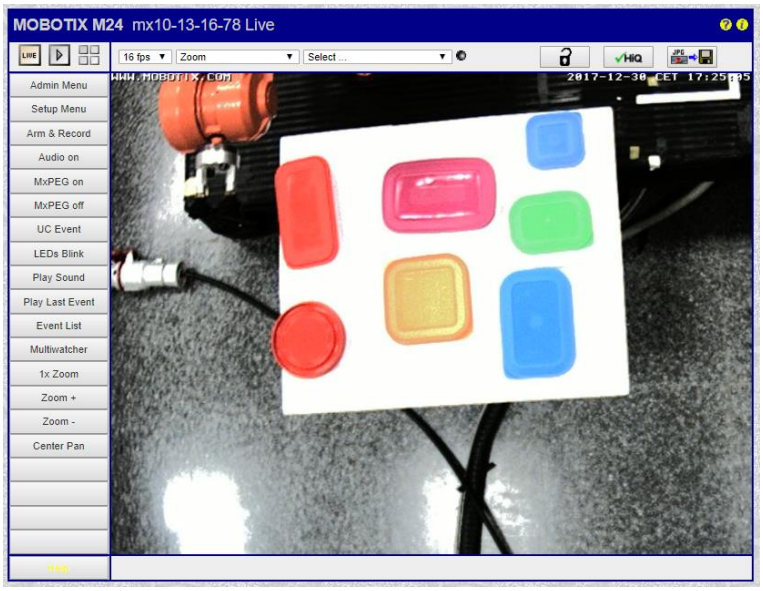
Per tant, les solucions escollides envers la distribució d'espais queda reflectida en el següent diagrama:



Il·lustració 10 Espai de desenvolupament del projecte

4.3. Càmera web

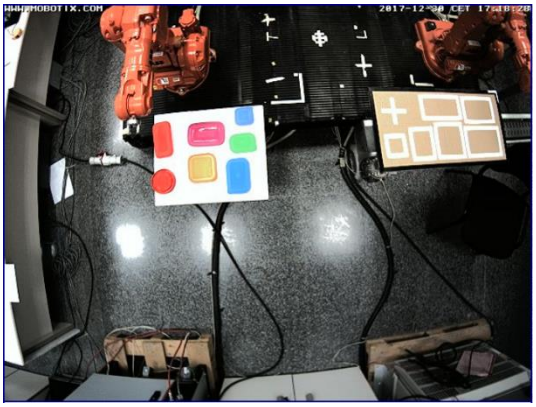
La càmera web que es troba al laboratori és una càmera amb accés via la seva adreça IP. Quan s'accedeix a aquesta IP el navegador mostra una pàgina de configuració d'aquesta amb multitud d'opcions de manipulació d'aquesta. Les funcionalitats que ens interessen pel projecte són dues en concret: el zoom i l'obtenció de la imatge en un instant concret de sol·licitud.



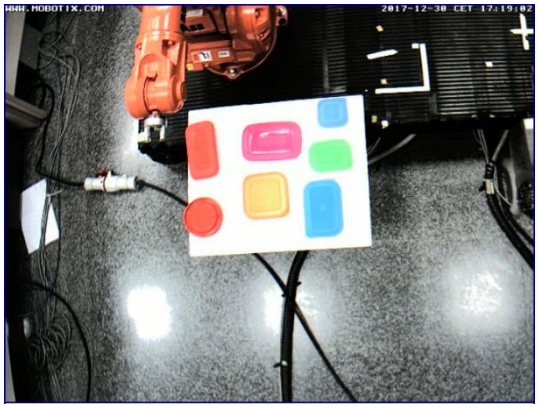
Il·lustració 11 Finestra per configurar paràmetres de la càmera web

4.3.1. Zoom

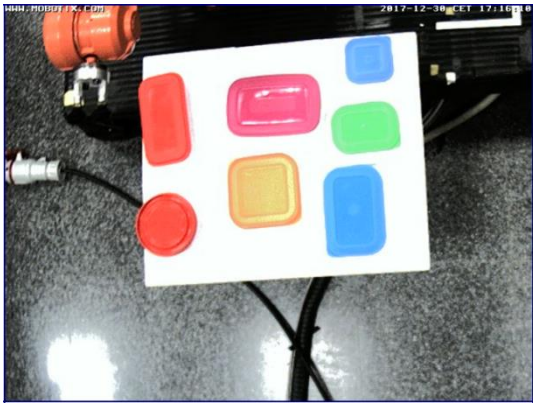
Les opcions de zoom de la càmera que utilitzem tenen un rang prou ampli de treball, amb diferents mesures preestablertes i facilitat per definir l'àrea d'interès a ser ampliada.



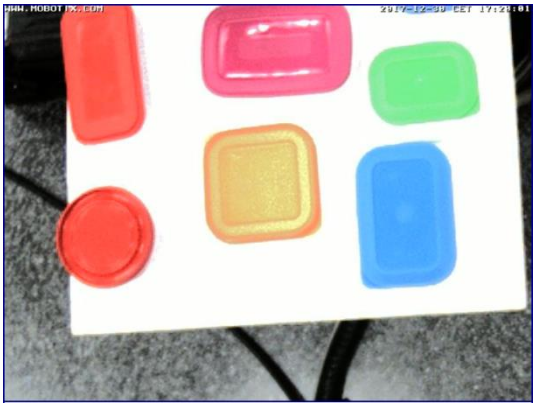
Il·lustració 12 Zoom mínim



Il·lustració 13 Zoom intermedi



Il·lustració 14 Zoom escollit



Il·lustració 15 Zoom màxim

Intuïm que en previsió de facilitar l'exactitud en la ubicació de les peces ens interessa fer un augment només sobre la zona aproximada on se situarà la taula, deixant un marge prou ampli com per a no haver de fer marques exactes de la posició d'aquesta en cada repetició de l'exercici. En avaluar les opcions d'augment i de centrat de l'àrea augmentada sobre la taula de treball escollida veiem que fent servir la tercera opció de zoom reduïm la quantitat de soroll a eliminar de la imatge però amb certa maniobrabilitat de col·locació d'aquesta, fent servir per tant la visió en la Il·lustració 13.

4.3.2. Obtenció d'imatge

La càmera web es troba inicialment connectada per cable a l'ordinador vinculat a la controladora del robot. Com aquest ordinador no disposa de connexió a Internet i la càmera tampoc es troba connectada a un router, s'elimina la possibilitat de connectar-se a aquesta per WiFi, provocant per tant que l'accés sigui necessàriament per cable.

Experimentant amb la connectivitat observem que, per establir comunicació amb la IP fixa de la càmera, s'ha de configurar el port de connexió per Ethernet de forma que només demani accés a la seva IP concreta, ja que provant d'establir contacte amb configuració de cerca d'IP automàtica no es rebia resposta.

Una característica que ens podia interessar de la interfície de la càmera és que la imatge que està rebent la càmera en cada instant pot ser accedida a través d'un enllaç directe.

A partir d'aquesta informació, la solució escollida és definir el port d'Ethernet per dedicar-lo exclusivament a la comunicació amb la càmera, i fer servir l'enllaç *http://10.13.16.78/record/current.jpg* per accedir a la imatge en directe en el moment de començar l'exercici.

4.4. Trasllat d'informació entre llenguatges

Les possibilitats de compartir la informació obtinguda al primer programa per utilitzar-la al segon programa, el gestor de moviments del robot, són fer-ho de forma automatitzada o manual.

La possibilitat de fer un procés automatitzat es dóna mitjançant un servidor de dades comú anomenat OPC.

OPC (*OLE (*Object Linking and Embedding*) for Process Control*) és un estàndard de comunicació fabricat per Microsoft la funció del qual és permetre la comunicació entre diversos llenguatges de programació permetent llegir i escriure variables compartides en un servidor comú.

En el nostre cas, la controladora del robot funciona com a servidor OPC.

Aquest mètode requereix que els compiladors dels programes que volen compartir informació estiguin connectats a un mateix servidor. També cal tenir nocions de programació d'un nivell mitjà-alt per escriure el codi que permet fer l'enviament de dades.

El mètode manual consisteix a preparar les dades que es volen intercanviar i guardar-les en un document a part per després copiar-les a l'altre programa.

El mètode que hagués estat idoni per aquest projecte era automatitzar el trasllat d'informació, però finalment, a causa de diversos inconvenients en l'aplicació d'aquest, s'ha escollit fer servir el mètode manual.

5. Desenvolupament de la solució escollida

El procés d'anàlisi de la solució que s'ha treballat es presenta ordenada en la direcció de processament d'informació que segueix el sistema, és a dir, es presentarà en primer lloc l'obtenció de la imatge, seguidament el processament d'aquesta, a continuació l'obtenció de dades necessàries per ser facilitades al robot i per últim el programa que reprodueix la imatge original.

5.1. Codi en C#: tractament d'imatge i obtenció de dades

El primer gran apartat a analitzar és el que s'ha programat en llenguatge C#, que comprèn el procés des de la consulta de la imatge on es veu la figura a reproduir fins a l'obtenció d'un arxiu amb els quatre paràmetres necessaris de cada objecte observat a la imatge: posició en l'eix X en mm respecte al centre de coordenades del robot, posició en l'eix Y en mm respecte al centre de coordenades del robot, orientació de l'objecte en graus respecte a l'eix X de la imatge i número identificador del tipus d'objecte segons color i forma.

El codi complet en C# està estructurat de la següent forma:

Declaració d'espai de noms

Es defineixen les llibreries que es faran servir i s'assignen referències ambigües de classes compartides en dues llibreries

Namespace PFG1_0

Contenidor on es desenvolupa el programa

Classe Pública Formulari

Finestra amb els botons, imatges inicial i sol·licitada i mostra de dades resultants

Variables Públiques generals

Classe Inicialització component

Botó 1: Inici del programa

Dona forma a la finestra i significat a l'acció de clicar el botó del pas 1

Botó 2: Final del procés

Mostra resultats

Mostra d'imatge rebuda

Assigna imatge rebuda a quadre d'imatge 1

Mostra d'imatge tractada

Assigna imatge tractada a quadre d'imatge 2, crida la funció de guardat de

Recepció d'imatge de la càmera

Detecció de colors

Filtre blanc i negre

Filtre inversor de colors

Eliminació de blobs petits

Classe de processament de dades general

Guardat de dades en document extern

Funció auxiliar

5.1.1. Llibreries AForge.NET

AForge és un marc de treball de codi obert C# dissenyat per a investigadors i desenvolupadors en els camps de visió per computació i intel·ligència artificial. Durant el desenvolupament del projecte han estat utilitzades dues de les llibreries que ofereix, ja que faciliten enormement el processament d'imatge. Aquestes llibreries són AForge.Imaging i AForge.Math.

AForge.Imaging, la seva llibreria més gran, està destinada a la manipulació d'imatge en pràcticament tots els seus aspectes. Podem trobar doncs que amb poques línies de codi es fan canvis en colors, llum, morfologia de la imatge, processament de píxels, filtratge de les dades d'una imatge per diversos mètodes matemàtics, etc.

AForge.Math, en canvi, conté diversos algoritmes matemàtics destinats a identificar formes geomètriques predeterminades a partir de la successió de píxels.

Aquestes llibreries han estat importades gràcies al configurador d'extensions de paquets NuGet, que és un repositori amb accés a milers de llibreries i facilita enormement la inclusió d'aquestes en el nostre programa, no havent de gestionar la descàrrega i inclusió manual d'aquestes.

Durant l'anàlisi del codi es parlarà en diverses ocasions de rutines pertanyents a aquestes llibreries, explicant la seva funció en cada cas.

5.1.2. Declaració de l'espai de noms

A la capçalera del programa trobem la declaració de les llibreries que es fan servir durant aquest, que són les pròpies de C# i les dues pertanyents a AForge comentades anteriorment, AForge.Imaging i AForge.Math. També podem observar dues assignacions, dels mètodes "Image" i "Point", ja que ambdós existeixen en dues llibreries diferents i en compilar ens detecta que hi ha un error de referències ambigües. Dessignem que "Image" pertany sempre a la llibreria del sistema i que "Point" pertany a la llibreria d'AForge.

(Annex 1 Línia 1) (Alt+Esquerra per tornar)

5.1.3. Namespace

El *Namespace*, o contenidor de noms únics, és el bloc que engloba tot el nostre programa i disseny de finestra.

(Annex 1 Línia 23) (Alt+Esquerra per tornar)

5.1.4. Classe pública del Formulari

El que al programa s'anomena formulari és la finestra o interfície que l'usuari veu en iniciar el programa, i des d'on es presenten les imatges i hi trobem els botons que sol·liciten tasques a realitzar sobre aquestes imatges. És per tant on es troben totes les parts del programa.

A l'inici es defineixen variables públiques, que són generals per a totes les classes. Aquestes són els quadres on es presenten les imatges a la finestra, l'*struct* objecteTrobat, que és un objecte on es poden guardar diverses variables, en el nostre cas els quatre tipus de dades que volem extreure de cada objecte (x, y, alpha i color); diverses variables per guardar imatges durant el processament d'aquesta i les rutes de directoris que es fan servir per guardar les dades.

A continuació es troben les classes i funcions que es faran servir.

(Annex 1 Línia 26) (Alt+Esquerra per tornar)

5.1.5. Inicialització de component

El primer apartat que trobem dins del formulari és la inicialització d'aquest. És una línia que el programa porta per defecte, ja que la seva tasca és inicialitzar la finestra en la qual treballarem per així poder llegir el seu contingut. En aquest apartat hem afegit línies per definir l'aspecte inicial de la finestra.

(Annex 1 Línia 52) (Alt+Esquerra per tornar)

5.1.6. Finestra d'interacció

L'interfície visual amb la que s'interactua per dur a terme el procés és senzilla i entenedora. És la següent:

Il·lustració 16 Finestra d'interacció amb el programa

S'hi poden observar dos botons, dos requadres per mostrar imatges amb etiquetes per esmentar el que s'hi veu i un quadre de text per presentar la solució.

El primer botó engega el processat de la imatge, i ens presentarà la imatge obtinguda per la càmera en el primer requadre i una presentació visual de les figures interpretades pel codi. També ens mostra el codi resultant dels càlculs.

El segon botó es prem en acabar la tasca robotitzada, i serveix per comparar la imatge resultant amb la obtinguda a l'inici del procés.

A continuació s'explica el codi que permet aquests resultats més detalladament.

5.1.7. Clic del primer botó

En el cas del programa escrit, l'ordre per iniciar el procés d'obtenció de dades queda definit per un esdeveniment, el clic del botó del primer pas.

En el moment que aquest botó és polsat, el primer succés és que els espais definits per a mostrar imatges es buida, ja que fent proves inicials vam detectar que si es pica diverses vegades el botó i no es neteja la imatge anterior, la memòria del programa emmagatzema en forma de dades cada imatge rebuda de la càmera. Si aquest fet es repeteix moltes vegades, s'alenteix el processament d'informació, i per tant el temps d'execució final.

A continuació es defineixen les coordenades on volem situar les imatges a la finestra de visualització, es fa una crida al programa que desenvolupa totes les tasques i s'assignen valors a dues etiquetes de la finestra de visualització on es mostra informació del que s'està visualitzant. També es presenten les dades que han estat emmagatzemades al document de dades final.

(Annex 1 Línia 63) (Alt+Esquerra per tornar)

5.1.8. Clic del segon botó

La funció del segon botó és presentar una comparativa entre la primera imatge obtinguda i el resultat final, representat pel robot. Per mostrar-ho, es fa una crida a la imatge inicial, guardada en una variable, i de nou a la classe encarregada de sol·licitar la captura en directe de la càmera web.

(Annex 1 Línia 92) (Alt+Esquerra per tornar)

5.1.9. Classe Mostra Imatge Rebuda

Fa una crida a la classe encarregada de sol·licitar la captura en directe de la càmera web i la mostra al primer requadre de la finestra.

(Annex 1 Línia 115) (Alt+Esquerra per tornar)

5.1.10. Classe Tractat i Càlcul

Aquesta classe és l'encarregada d'executar la resta de classes i funcions, les quals fan el processament de la imatge, l'extracció de les dades que necessitem per replicar la imatge i el guardat d'aquestes dades en un fitxer extern.

L'execució ordenada modifica funció rere funció una mateixa imatge. En primer lloc es simplifiquen els colors d'aquesta i se'n guarda una còpia per ser consultada posteriorment. Tot seguit es detecten les formes dels envasos per obtenir la posició i orientació de cadascun d'aquests. Finalment, s'envien totes aquestes dades a un fitxer extern, i es retorna a la finestra de visualització el resultat final del processament en dues modalitats: visualment sobre la imatge inicial, i en forma de dades numèriques.

(Annex 1 Línia 129) (Alt+Esquerra per tornar)

5.1.11. Comunicació amb la càmera web

L'objectiu de la classe és establir comunicació amb la direcció web on es pot consultar l'última imatge presa per la càmera. El procés comunicatiu crea un buffer on emmagatzemar dades, demana a la direcció web accés a aquesta i en rebre una resposta positiva segons la web està en funcionament fa una lectura de cada byte de dades i els emmagatzema ordenadament al buffer. En acabar, es transfereixen aquestes dades a un mapa de bits, que és un format d'imatge on es presenta la informació rebuda en forma de fotografia formada per píxels.

En obtenir la fotografia de l'estat actual de la visió de la càmera, es retorna a la variable de la finestra encarregada de mostrar la imatge.

(Annex 1 Línia 162) (Alt+Esquerra per tornar)

5.1.12. Detecció de colors

Com s'ha comentat abans, un dels objectius del programa és identificar el color dels objectes que es presenten per tal de distingir-los entre sí. Aquest apartat del programa fa una lectura de cada píxel de la imatge, classificant si el seu color pertany a algun dels rangs que volem identificar.

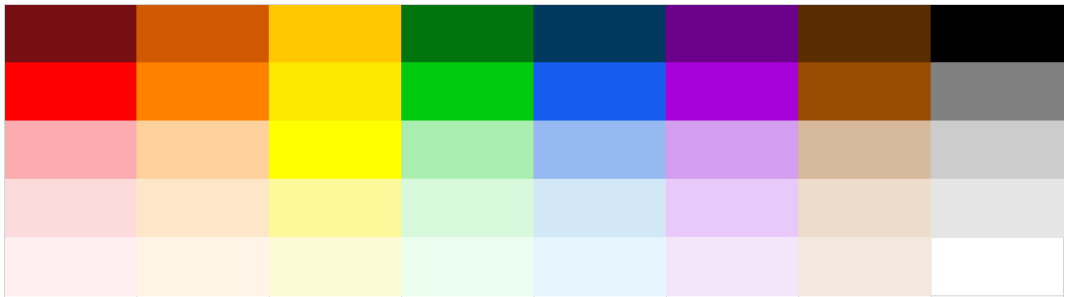
Cada píxel de la imatge guarda la informació del seu color en format RGB (Red Green Blue, o vermell verd blau).

RGB és un model de color en què cada tonalitat de l'escala cromàtica es defineix a partir de l'adhesió de tres colors primaris de llum. Cadascun dels tres colors es defineixen en un rang de números que, en el nostre cas, va de 0 a 255. Els números identifiquen la quantitat de llum que aporta aquest color a la barreja, trobant que un valor de zero no aporta cap quantitat d'aquest color a la barreja i 255 aporta la màxima quantitat de color.

Per exemple, si busquem l'identificador RGB del color vermell, ens trobarem que els seus valors són (R, G, B) = (255, 0, 0), ja que el color vermell de la combinació té la màxima intensitat possible i els altres dos colors primaris no aporten cap informació a la barreja, esdevenint així el color vermell pur.

Amb el mateix exemple, si l'identificador RGB que trobem és (125, 0, 0), podem suposar que el color definit és un vermell fosc, ja que la barreja només consta de vermell però la quantitat de llum serà la meitat que en el cas anterior, o el que a la vista és el mateix, la barreja de vermell amb una petita quantitat de color negre.

A continuació es pot observar una taula amb alguns exemples de colors identificats en format RGB:



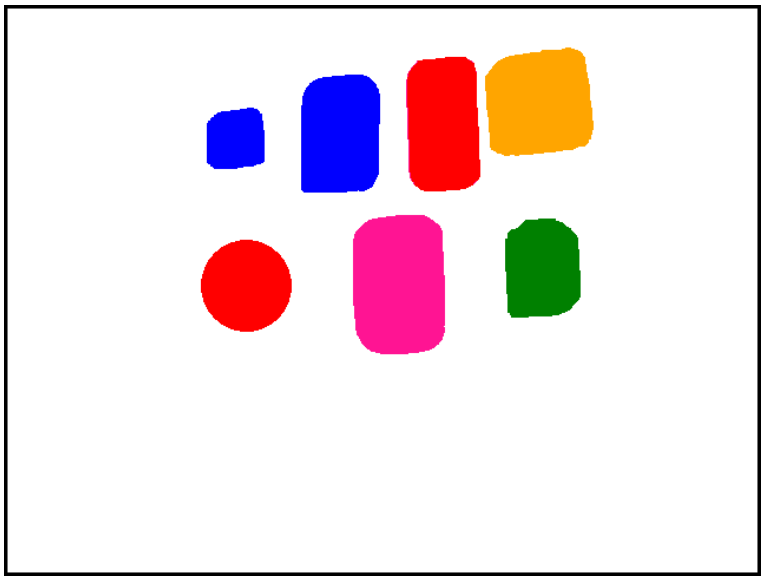
119,14,18	208,88,0	255,200,0	0,117,13	0,57,94	108,1,139	90,45,1	0,0,0
255,0,0	255,129,0	255,233,0	0,202,14	22,93,239	166,1,214	151,76,2	128,128,128
252,171,174	253,208,156	255,255,0	169,238,174	149,185,240	211,158,240	215,185,156	205,204,204
253,218,219	254,230,201	252,249,155	217,249,221	211,232,246	233,201,250	238,220,202	229,229,229
254,238,240	255,244,230	252,250,211	236,254,239	230,245,254	243,229,250	242,232,222	255,255,255

Il·lustració 17 Taula cromàtica amb identificadors en format RGB

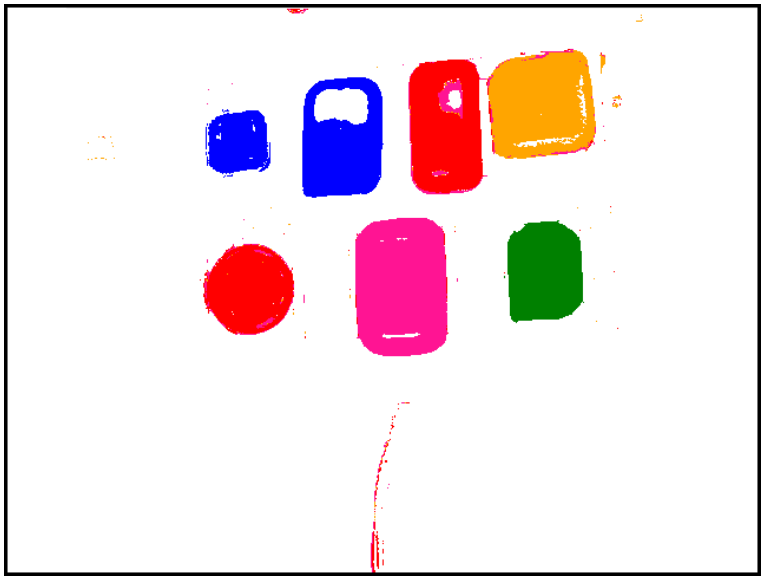
Partint d'aquests coneixements, el propòsit d'aquesta funció és descartar els rangs que no ens interessen i unificar els que ens aporten informació. Els colors que ens interessa identificar són els pertanyents a les tapes dels envasos, que són el blau (rectangle petit i rectangle gran), el vermell (rectangle i rodona), el verd, el rosa i el taronja. A partir de definir una il·luminació adequada i invariant a l'aula, la imatge obtinguda, diverses consultes en taules de referència i assajos de prova i error s'obtenen els rangs de colors definits en format RGB que ens interessen.

Es determina que en la consulta de cada píxel s'avaluarà si pertany a algun dels rangs. En cas afirmatiu el píxel es canvia de color retornant tots els pertanyents a un rang amb un mateix color. En cas negatiu el color del píxel canvia blanc, retornant amb això una imatge bastant simplificada cromàticament, però amb algunes coincidències de color que no pertanyen als envasos, que d'ara endavant anomenarem "soroll".

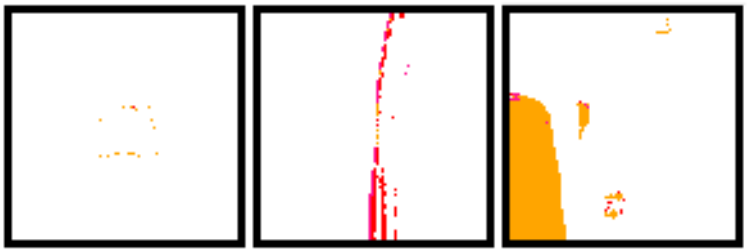
Per culpa de reflexos en el plàstic provocats per la situació invariable de llum artificial, és possible trobar que algunes parts dels envasos es converteixen en taques blanques com si es tractés d'un forat. Per sort, en general aquestes taques queden situades als marges dels envasos, i no al centre on en passos següents consultarem el color.



Il·lustració 18 Resultat ideal processament de colors



Il·lustració 19 Resultat del processament de colors



Il·lustració 20 Detall del soroll no processament

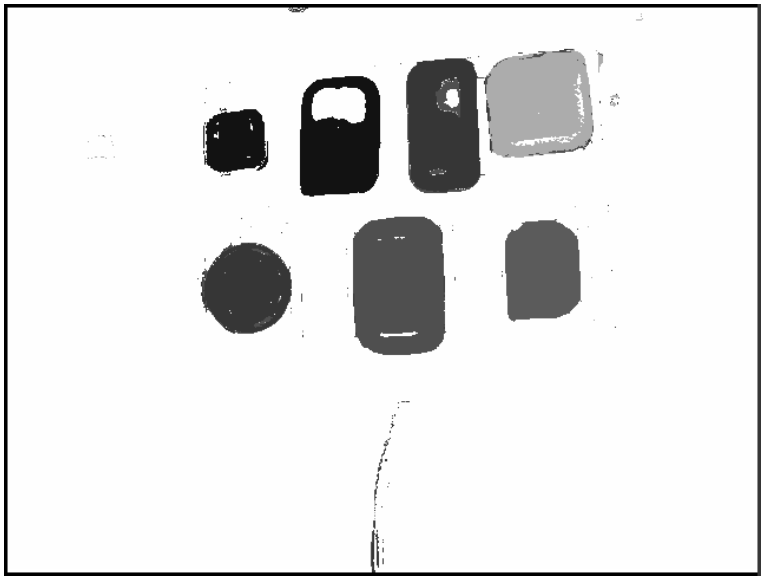
Es retorna la imatge tractada, i es guarda en una variable nova per conservar-la i poder-se consultar més endavant.

(Annex 1 Línia 198) (Alt+Esquerra per tornar)

5.1.13. Filtre Blanc i Negre

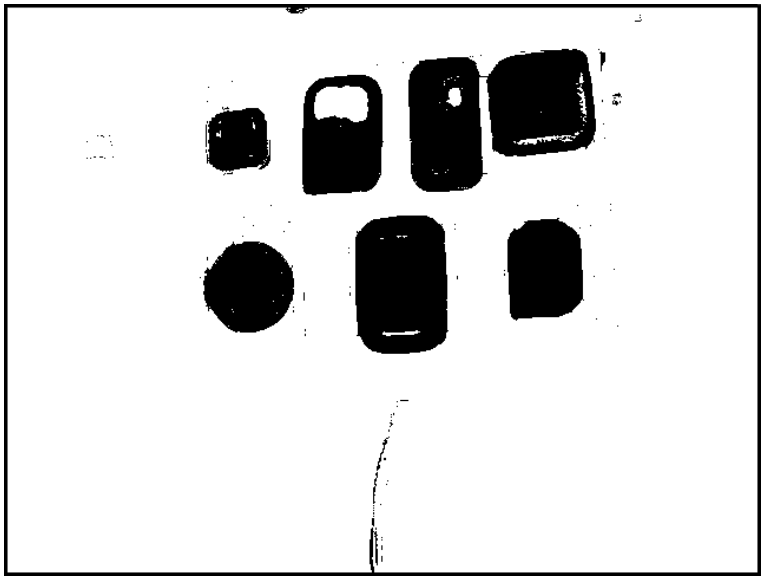
El procediment de passar una imatge en color a blanc i negre es diu binarització, ja que es decideix entre dues opcions de color, el blanc o el negre. Aquest procediment es complica si es vol fer directament a partir d'una imatge en color, ja que seria complex definir els criteris pels quals un color es defineix com a blanc o com a negre durant el canvi. És per això que abans de binaritzar cal un pas previ, que es tracta de passar la imatge en color a una imatge en escala de grisos.

L'escala de grisos està predeterminada amb tonalitats de grisos assignats a la lluminositat dels colors. El resultat ens retorna una imatge amb un sol color en diverses lluminositats.



Il·lustració 21 Imatge en escala de grisos

A partir d'aquesta imatge, es fa un filtratge *threshold*, o filtratge límit. La seva funció és definir un valor de lluminositat per sobre del qual es transformaran els grisos en blancs, i per tant, la resta en negres. Amb aquest filtre, i partint de la nostra imatge on només trobem fons blanc amb soroll i conjunts dels colors dels envasos, obtenim una imatge amb fons blanc, soroll en forma de píxels individuals i unes taques negres grans situades on es troben els nostres envasos de colors. Els possibles reflexos provocats per la llum artificial els observarem en forma de taques blanques petites dins de la silueta negra dels envasos.

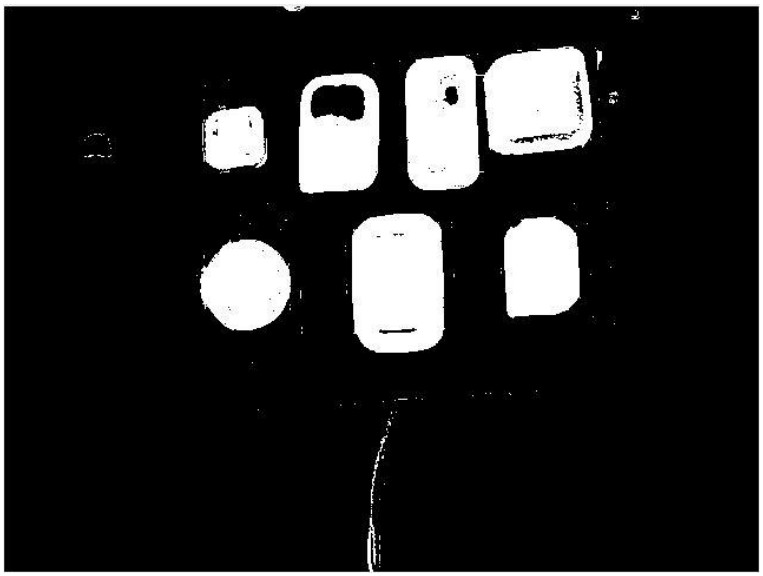


Il·lustració 22 Imatge en blanc i negre

(Annex 1 Línia 247) (Alt+Esquerra per tornar)

5.1.14. Filtre inversor

El filtre inversor simplement converteix els píxels blancs d'una imatge binaria en píxels negres, i els píxels negres en blancs.



Il·lustració 23 Imatge B/N amb filtre inversor

(Annex 1 Línia 260) (Alt+Esquerra per tornar)

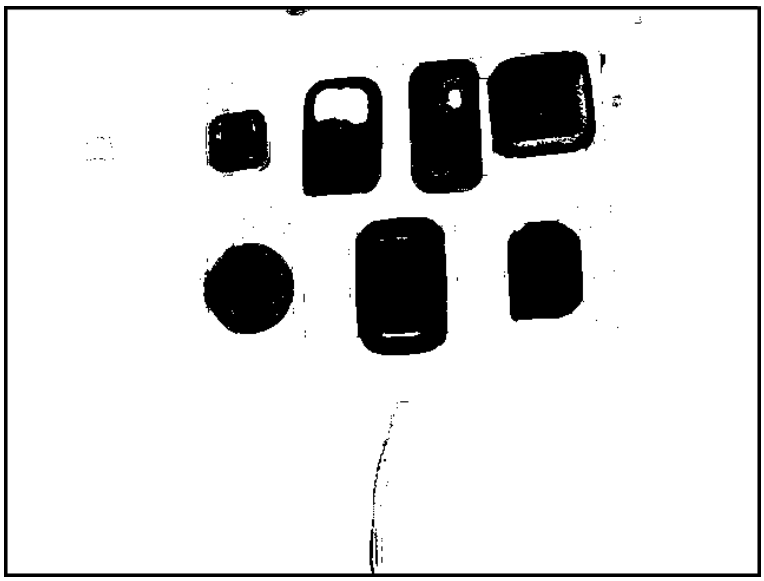
5.1.15. Filtre de blobs petits

La paraula *blob* denomina un conjunt de píxels connectats que comparteixen una mateixa característica. Buscant els límits dels conjunts de píxels es poden definir blobs com a ens unitaris, taques que es poden processar com a objectes individuals.

Aquesta classe fa servir una funció pertanyent a la llibreria AForge destinada a localitzar blobs en una imatge i filtrar-los segons els paràmetres que se li indiquen. Per defecte, la funció suposa que la imatge que se li dóna serà un fons negre amb blobs blancs. Les opcions a definir possibles són l'amplada mínima i màxima i l'alçada mínima i màxima. En el nostre cas, se li determina una amplada i alçada mínimes una mica menors que les dimensions de l'envàs verd, ja que és el més petit.

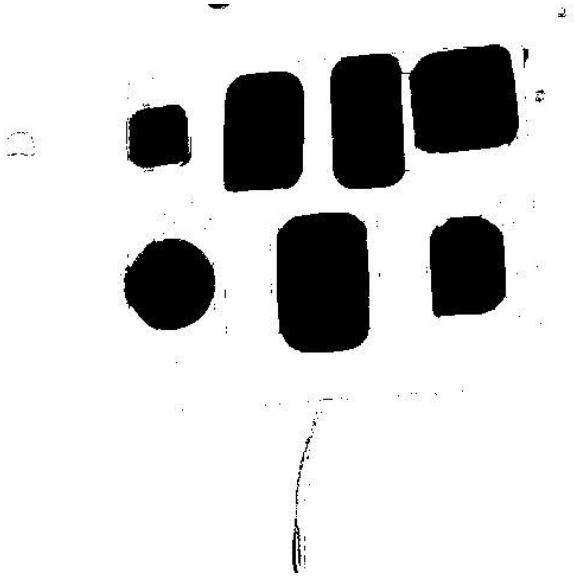
Aquest filtre es fa servir dos cops, gràcies a la facilitat per eliminar imperfeccions de la imatge final que volem obtenir.

En primer lloc tenim una imatge de fons blanc amb soroll en forma de píxels negres, i blobs negres amb imperfeccions blanques a dins pertanyents als reflexos de la superfície brillant de les tapes.



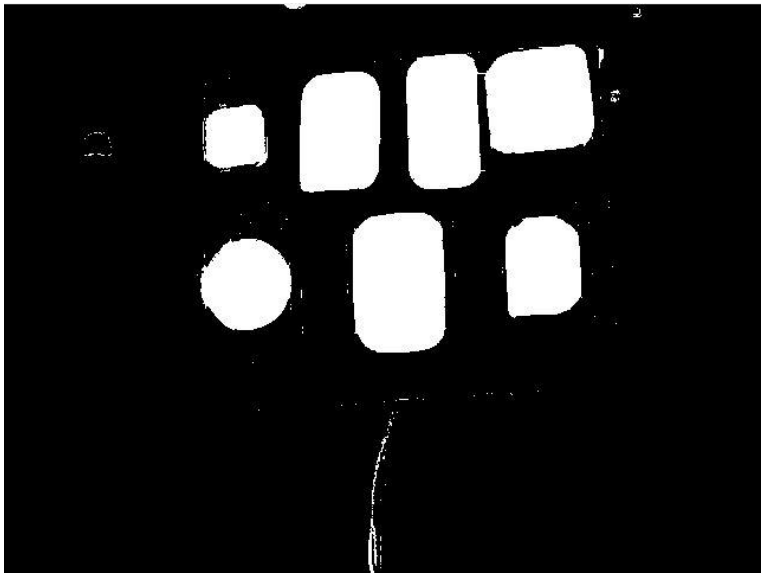
Il·lustració 24 Imatge B/N a l'inici del procés d'eliminació de blobs

En filtrar aquesta imatge obtenim l'eliminació de les imperfeccions creades pels reflexos de llum, però no el soroll.



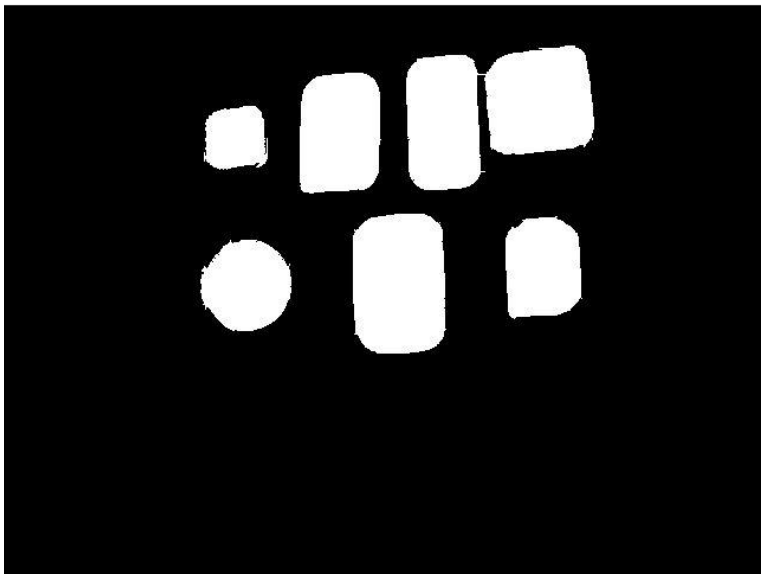
Il·lustració 25 Imatge B/N amb soroll sense reflexos

Invertim la imatge, aconseguint un fons negre amb soroll i blobs sòlids de color blanc.



Il·lustració 26 Imatge B/N sense reflexos invertida

En tornar a filtrar la imatge, el soroll desapareix en detectar-se com blobs petits, aconseguint així una imatge de fons negre que conté únicament blobs blancs sòlids.



Il·lustració 27 Imatge B/N sense soroll ni reflexos

Per últim, es retorna la imatge en aquest estat.

(Annex 1 Línia 273) (Alt+Esquerra per tornar)

5.1.16. Eliminació de soroll extra

El filtre *Median* que s'aplica a la imatge en blanc i negre s'assegura de suavitzar les vores de la silueta dels envasos, ja que canvia el valor dels píxels en funció del color predominant al seu voltant. En el nostre cas el fem servir al final del procés de canvi a blanc i negre, ja que a causa de reflexos de l'enllumenat sobre la taula, en algunes ocasions identificava dos blobs diferents com a un de sol a causa d'una línia d'un sol píxel que els unia.

(Annex 1 Línia 292) (Alt+Esquerra per tornar)

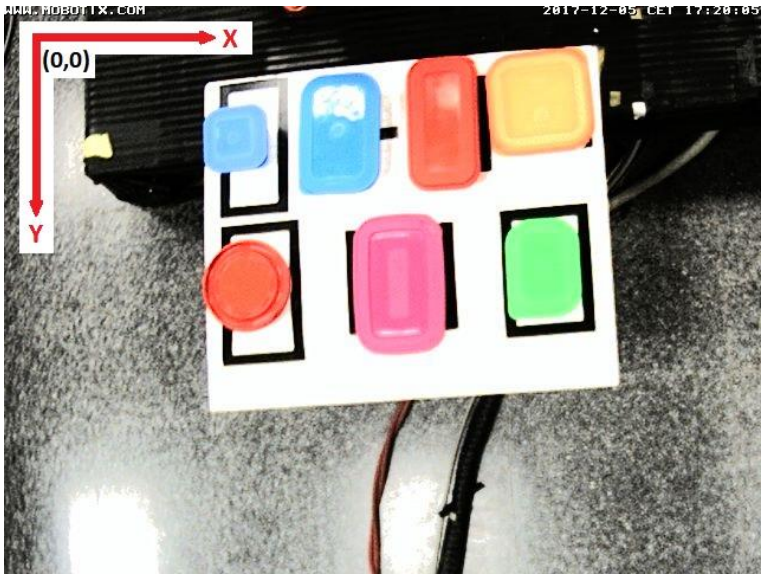
5.1.17. Obtenció de formes i mesures dels blobs

Aquesta classe és la que més tasques realitza, ja que permet la majoria de càlculs que hem de fer per obtenir dades definitives. S'explica la primera part de la classe Cos del Procés Principal.

Abans d'analitzar-la però, cal explicar que per facilitar el traspàs de dades a una taula s'han assignat nombres enters d'1-7 per distingir els set tipus d'envasos que podem trobar sobre la taula. Els nombres assignats són els que segueixen:

1 = Rosa
2 = Taronja
3 = Verd
4 = Blau petit
5 = Blau gran
6 = Vermell rectangular
7 = Vermell rodó

També s'ha de tenir en compte la disposició dels eixos de coordenades com les utilitza el compilador. En aquest cas, la imatge obtinguda ja sigui en format *Image* o *Bitmap* (objecte utilitzat per treballar amb imatges definides per dades de píxel) considera que el centre de coordenades (0,0) està situat a la cantonada superior esquerra d'aquesta. L'eix horitzontal es considera l'eix de les X, en sentit positiu d'esquerra a dreta, i l'eix vertical és l'eix de les Y, en sentit positiu de dalt a baix, en unitats píxel:



Il·lustració 28 Eixos de coordenades imatge

A l'inici es declara una estructura múltiple que volem omplir amb les dades de color, posició i orientació respecte als eixos dels envasos trobats a la imatge.

Es fa servir un nou filtre pertanyent a la llibreria d'imatge d'AForge, la funció del qual és comptar el nombre de blobs que es troben a la imatge i agafar-ne les mesures. A continuació, dins d'un bucle es processa cada blob.

La primera comprovació, gràcies a una funció de la llibreria matemàtica d'AForge, determina si la forma geomètrica del blob és una circumferència. En cas positiu se n'obté el radi i el centre, es guarden les coordenades X i Y del centre de la circumferència a la taula, s'assigna al valor *alpha* un zero, ja que ens és igual l'orientació si es tracta d'una circumferència, i s'assigna directament el valor 7 a la dada que guarda el color, ja que no hi ha altres envasos rodons amb el que confondre's.

En cas de trobar que el blob té la forma de qualsevol altra figura, que en aquest cas seran polígons aproximadament quadrangulars, s'extreuen les dades dels vèrtexs d'aquest en coordenades X i Y.

La documentació de la llibreria ens indica que la funció detectora de blobs quadrangulars guarda una llista amb les coordenades de les cantonades de cada blob, però també indica que no estan ordenades en cap sentit. És per això que ens calen càlculs matemàtics vectorials per determinar tant les coordenades X i Y centrals del quadrilàter, que guardem a la taula com a coordenades de posició, com per guardar en variables separades les coordenades de cadascun dels vèrtexs, distingint si pertany a la cantonada superior, inferior, la de mes a la dreta o a l'esquerra. Això s'obté comparant entre sí els diferents valors de X i Y de la llista de vèrtexs.

Fent ús d'aquestes dades, també es determina l'orientació del polígon. Es realitza un càlcul per aïllar l'angle alpha, en graus, d'un dels costats de l'envàs respecte de l'eix de les X de la imatge, concretament el costat connectat pels vèrtexs que es troben més a l'esquerra i més a baix.

L'angle que ens interessa obtenir per saber l'orientació de l'envàs és el format entre el costat més llarg d'aquest i l'eix X de la imatge, però amb aquest mètode no determinem de quin costat es tracta. Comparant la longitud de dos costats adjacents podem sumar 90 graus si l'angle calculat pertany a un costat curt.

Per últim aquest angle es guarda com a valor *alpha* de la figura detectada, sempre amb un valor entre 0 i 180 graus.

Amb l'objectiu de trobar l'última dada, que és el color de l'envàs que està essent analitzat, i fent ús de les coordenades centrals del polígon, es consulta en aquesta mateixa posició el color de píxel observat a la imatge de colors simplificats obtinguda a l'apartat 5.1.11.

Si els valors RGB coincideixen amb els colors rosa, taronja, vermell o verd s'assigna a la variable *color* el número corresponent.

En cas de trobar que els valors RGB coincideixen amb el color blau, s'ha de discernir si es tracta de l'envàs blau petit o el gran.

En aquest cas es dedueixen les longituds de dos costats del polígon continus a partir de les cantonades conegudes, es calcula l'àrea del polígon i s'assigna el valor d'envàs blau petit o envàs blau gran a partir d'aquesta dada.

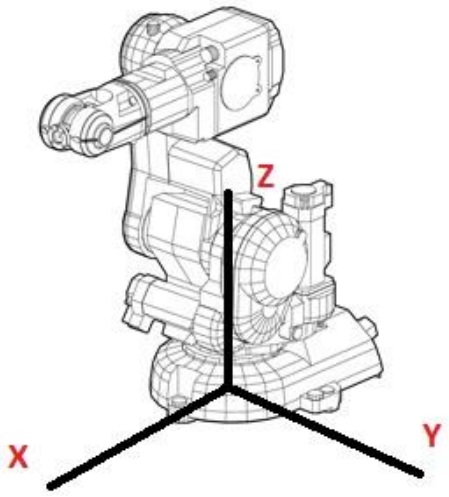
Finalment es guarden a una taula els quatre valors recopilats (posició X central, posició Y central, angle alpha i nombre enter identificador de color) i es posen a zero totes les variables per repetir el procés amb el següent blob.

(Annex 1 Línia 303) (Alt+Esquerra per tornar)

5.1.18. Canvi de coordenades

S'explica la segona part de la classe Cos del Procés Principal.

Les coordenades del robot, que vénen donades per la documentació d'aquest, situen el centre de coordenades d'aquest a la base del robot i al centre d'aquest. Trobem definits tres eixos, X, Y i Z, considerant l'eix X el que parteix horitzontalment cap a la situació inicial del robot, apuntant endavant; l'eix Y és l'eix que parteix horitzontalment en direcció lateral i a 90 graus de l'eix X; i l'eix Z apuntant amunt, considerant l'alçada a la qual es troba l'extrem d'aquest respecte de la base del robot:



Il·lustració 29 Eixos de coordenades robot

Els càlculs per transformar les coordenades de la imatge a coordenades robot comencen considerant que els eixos horitzontals són els mateixos però amb el nom canviat, trobant que el que a la imatge considerem eix X al robot l'anomenen eix Y. Aquesta assumpció no és del tot correcta, ja que la càmera no està situada just a sobre de la taula sinó aproximadament sobre la cantonada més allunyada del centre de coordenades robot de la taula, però fa una aproximació amb un error màxim d'uns ±20 mm.

Partint d'aquesta consideració, es fan càlculs de proporcionalitat entre les coordenades d'ambdós sistemes.

Agafant dos punts de referència als extrems de la taula, s'adquireixen les dades de coordenades de la imatge, i posicionant el robot en aquests punts es guarden les coordenades del robot en el mateix punt. Obtenim la següent taula:

Posició de referència	Coordenades Píxel	Coordenades robot
1	X = 456 Y = 264	X = 701.13 Y = 382.30
2	X = 190 Y = 82	X = 352.40 Y = -64.35

Les equacions de la relació de proporció, considerant el canvi de nom entre eixos, són:

$$X_{robot} = a \cdot Y_{imatge} + b$$

$$Y_{robot} = a \cdot X_{imatge} + b$$

Desenvolupant un sistema de dues equacions amb les coordenades X del robot dels dos punts de referència, trobem que la transformació per trobar la X de la nova peça detectada serà:

$$X_{robot} = 1.645 \cdot Y_{imatge} + 288.54$$

I seguint el mateix mètode, trobem l'equació de transformació per la nova Y:

$$Y_{robot} = 1.85173 \cdot X_{imatge} - 482.4136$$

Amb aquestes dades, i agafant la informació de la posició en coordenades de la imatge, la classe ens retorna les coordenades on el robot s'ha de dirigir a dipositar l'envàs detectat.

(Annex 1 Línia 505) (Alt+Esquerra per tornar)

5.1.19. Funció de guardat de dades

En el procés de guardat de dades s'envien aquestes a un document extern, el qual serà traslladat al segon ordinador per desenvolupar la tasca robotitzada.

El procés crea o modifica al directori assignat un document de text (.txt) anomenat Dades. En ell s'escriuen les funcions que guarden variables en llenguatge RAPID d'ABB. Això facilita el trasllat d'informació, ja que enganxant aquestes línies de codi al programa del robot no es produeixen errades de lectura per part de la controladora.

(Annex 1 Línia 545) (Alt+Esquerra per tornar)

5.1.20. Classe auxiliar

Al final del codi en llenguatge C# podem trobar una classe auxiliar que assigna els valors necessaris a la variable *points* utilitzada en el processament d'informació de posicions.

(Annex 1 Línia 629) (Alt+Esquerra per tornar)

5.2. Traspàs d'informació

El document resultant del guardat de dades s'emmagatzema a la memòria USB connectada a l'ordinador. Un cop executat el primer programa es trasllada la memòria al segon ordinador. El codi amb les dades es copia al programa encarregat de gestionar els moviments del robot, i com aquest ordinador té accés als directoris de la controladora, es guarda a la carpeta del nostre projecte, preparat per ser executat.

5.3. Codi en RAPID, moviments del robot

El codi escrit en RAPID és més senzill que l'escrit en C#, ja que les tasques que ha de realitzar són similars entre elles. El moviment de dipositat dels envasos és quasi el mateix, només diferenciant el punt d'inici de recorregut depenent de l'envàs a ser col·locat.

El funcionament esperat d'aquest és que es rebin les dades X i Y en coordenades del robot, l'angle *alpha* en graus i el color en forma de nombre enter com a identificador, i que s'apliqui el moviment que disposa cada objecte a la seva posició amb l'orientació adequada.

5.3.1. Declaració de configuracions del robot

A l'inici del programa es troben declarades diverses posicions fixes en forma de *robtarget.* Les variables *robtarget* són un tipus de contenidor de dades preparades per emmagatzemar tots els descriptors d'un punt en l'espai, com són la posició en coordenades (X, Y, Z), l'orientació de l'eina de l'extrem en quaternions, el tipus d'eina que subjecta, etc.

Com es comenta anteriorment, la zona de magatzem dels envasos que hauran de ser agafats té una posició fixa, i per tant trobem una variable de posició per a cada configuració de recollida d'envàs.

Les posicions s'han anomenat amb el nom del color i/o forma que els caracteritza (p.ex. robtarget verd, robtarget rodo (envàs de forma rodona), robtarget blaup (blau petit), etc.).

A continuació de cada configuració s'ha duplicat aquesta mateixa però augmentant la coordenada Z (coordenada de l'alçada) en 200 mm, de forma que abans d'accedir a la posició de recollida cal, obligatòriament, passar per la posició de "seguretat" de cada envàs i reduir l'alçada en línia recta. Amb aquest pas obligatori, evitem col·lisions entre els envasos durant el recorregut de col·locació.

Aquestes variables duplicades s'anomenen amb el nom del color i/o forma que els caracteritza precedit per una lletra *s*, indicant que es tracta de les coordenades de seguretat de l'envàs a recollir (p.ex. robtarget sverd, robtarget srodo (envàs de forma rodona), robtarget sblaup (blau petit), etc.).

També trobem definides altres configuracions de suport. Aquestes són les següents:

- Configuració *home*: Per la manca d'una bateria per conservar la memòria a la controladora del robot, cada cop que l'encenem ens avisa que els

compta-revolucions no estan actualitzats, i per tant no sap en quina configuració es troba el robot. Al laboratori està definida una configuració inicial que la fan servir totes les persones usuàries, i cada cop que es vol fer servir el robot s'actualitzen els compta-revolucions per permetre fer càlculs de trajectòria a la controladora. Aquesta configuració és la conservada amb el nom *home*.

- Configuració *repos*: aquesta configuració situa el robot en un punt sobre la zona de magatzem. Es fa servir com a punt d'espera a l'hora de sol·licitar una imatge de la càmera, evitant que el robot aparegui a la fotografia provocant una lectura errònia, i com a punt de pas entre col·locacions dels envasos, per evitar que mogui aquests o els que es troben a l'àrea de magatzem.

- Variables *posicio* i *sposicio*: Es tracta de les variables de tipus *robtarget* que es faran servir per indicar la posició i orientació de cada element que volem que sigui deixat a la taula, i el pas previ de seguretat per no arrossegar envasos ja dipositats.

  Els valors d'aquestes variables s'assignaran amb les dades rebudes del programa anterior cada cop que es vol situar un envàs.
  La X i la Y depenen de la transformació de coordenades imatge a coordenades robot obtingudes.
  La Z sempre és la suma de l'alçada de la posició de recollida amb la diferència d'alçada entre el magatzem i la taula, que en el nostre cas són 135 mm. Cada variable Z està definida dins de la rutina de posicionament de cada tipus d'envàs.

(Annex 2 Línia 1) (Alt+Esquerra per tornar)

5.3.2. Declaració de variables generals

En el programa cal declarar el valor de les variables X, Y, *alpha* i *color* abans de reproduir el programa. Es declaren aquestes variables com a constants numèriques amb un valor definit que hem obtingut del programa de processament d'imatge.

(Annex 2 Línia 25) (Alt+Esquerra per tornar)

5.3.3. Assignació de valors per a cada envàs

El valor màxim d'envasos a ser processats actualment es va decidir que fossin 10. Partint d'aquesta premissa, es fa un bucle de 10 repeticions en què en cada cas s'han assignat els valors d'un envàs a ser processat. En fer l'assignació es consulta el color i es posiciona l'envàs adequat, i seguidament es fa la lectura de les dades del següent procés.

5.3.4. Orientació dels envasos

Tot i la dificultat d'aplicar el sistema de càlculs per quaternions que fa servir el descriptor *robtarget*, s'ha trobat una manera més senzilla de resoldre l'orientació dels envasos respecte a un eix de referència.

Recordem que el programa ens facilita les dades de l'orientació de l'envàs en forma d'angle respecte a l'eix X de la imatge, i per tant, respecte a l'eix Y del robot.

Llegint la documentació completa del robot s'ha trobat una línia de codi que, rebent una orientació en forma d'angle, gira l'eix de l'extrem del robot el nombre de graus que se li assigna. A part d'això, s'ha fet una investigació de la resposta del mateix extrem aplicant una combinació de quaternions concreta. S'ha trobat que aplicant els quaternions de la forma (0,1,0,0) l'extrem es posiciona en posició vertical, amb les ventoses orientades de forma que es troben sobre un mateix valor de X. Partint d'aquesta posició, es demana al robot que abans de deixar l'envàs l'orienti amb el valor de la variable *alpha*.

5.3.5. Moviments del robot

La programació de moviments és senzilla, i depenent del valor que pren la variable *color* es desenvolupa una sèrie de moviments similars canviant únicament les dades de posició final i el tipus de peça recollida del magatzem.

Un cop identificat el color, la variable Z pren el valor d'alçada del punt on ha de deixar l'envàs. A continuació, s'assignen els valors X, Y, Z i alpha a les variables *robtarget* anomenades *posicio* i *sposicio*.

Partint de la posició de repòs on es troba el robot, sobre la zona de magatzem, aquest es situa sobre la peça a agafar. De la posició de seguretat de l'envàs a recollir descendeix 200mm fins a tocar-lo. En aquest moment s'activa l'eina de succió del robot i s'espera un segon.

Tot seguit es desplaça verticalment fins al punt de seguretat de nou, es dirigeix a les coordenades assignades per la variable *sposicio* i orienta l'envàs. Llavors descendeix a *posició*, es desactiva la succió de l'eina, espera un segon de nou i retorna a la posició de repòs sobre la zona d'emmagatzematge, passant abans per la posició de seguretat de la peça que s'acaba de deixar.

En acabar el moviment, el bucle de lectura procedeix a llegir les dades de la següent peça.

En els casos en què el valor de la variable *color* és 0 no es fa cap actuació. Aquest és l'indicador per saber que no cal processar cap element més.

(Annex 2 Línia 62) (Alt+Esquerra per tornar)

6. Proves i resultats

6.1. Procediments descartats

6.1.1. Il·luminació

El procés per treballar amb el reconeixement d'elements de la imatge ha estat difícil, degut principalment a dos factors: reflexos dels envasos i diferents il·luminacions ambientals.

Els envasos que es fan servir són reutilitzats d'altres projectes, i estan fets de plàstic amb un acabat sense rugositats. Aquest fet fa que la llum artificial del sostre reboti perfectament, i com que hi ha diverses fonts de llum al voltant que es controlen amb un mateix interruptor, no és possible apagar la que es troba més a prop de la zona de treball sense perdre massa quantitat d'il·luminació. Inicialment la configuració que es va escollir es tractava de mantenir tota la llum artificial del laboratori encesa tret de la línia que es troba vertical sobre l'àrea de treball, substituint-la per llum natural obrint les persianes més properes. El problema era que es depenia d'aquesta llum natural, eliminant la possibilitat de realitzar processos quan marxava el sol o feia mal temps.

Es va optar per aportar una altra llum artificial amb un llum portàtil, però la il·luminació que aportava era escassa.

L'última prova va ser encenent tota llum artificial, comptant amb què es tindrien els reflexos, però retocant paràmetres de saturació de color de la càmera. Aquesta ha estat la configuració definitiva d'il·luminació, sabent que en el processament de la imatge hi ha reflexos a ser tractats.

6.1.2. Classe eliminaSoroll

En els primers passos de processament de la imatge es va presentar la dificultat d'eliminar el soroll derivat del filtratge de colors. Partint de la teoria de tractament d'imatge, es va pensar que aplicar un filtre anomenat *Median Filter* es podia eliminar aquest.

El *median filter* és un tractament digital d'imatge que es fa servir sovint per a l'eliminació de soroll. El procediment d'aquest filtre substitueix el valor d'un píxel pel valor mitjà dels 8 píxels que l'envoltes, reduint la diferència entre aquests per retornar una imatge més suau.

En aplicar-lo, es va veure que el soroll de la imatge estava format per conjunts de píxels, i no per píxels solitaris. Aquest fet provocava que el *median filter* no retornés el resultat esperat, pel que es va decidir que no podia ser el mètode d'eliminació de soroll únic. Com s'ha vist, però, es fa servir en un altre pas del processament d'imatge.

6.1.3. Traspàs de dades via OPC

Durant la definició del projecte amb la tutora es va plantejar fer la comunicació entre programes fent ús d'OPC.

La idea principal era fer funcionar ambdós programes en un mateix ordinador, permetent el traspàs de dades amb OPC de forma que el sistema quedava totalment automatitzat, però una sèrie d'adversitats no han permès fer-ho possible.

El primer que es va intentar va ser instal·lar la plataforma Visual Studio a l'ordinador que té accés als directoris del robot i que permet activar rutines sense fer servir la Teach Pendant. La manca d'internet en aquest ordinador obligava a descarregar el programa en un altre dispositiu i instal·lar des d'un USB. Es va provar a instal·lar la versió de 2015, ja que és la mateixa que es fa servir a l'ordinador personal on s'ha treballat, però el sistema operatiu, que es tracta de Windows 7, comunicava que aquest programa requeria un sistema operatiu més recent. No es va poder instal·lar tampoc amb mode compatibilitat, i la resposta era la mateixa quan es van intentar instal·lar versions anteriors de Visual Studio (VS 2010 i VS 2008).

La següent opció valorada va ser fer una connexió per cable entre els dos ordinadors, però això obligava a desconnectar la càmera, fet que impossibilitava automatitzar el sistema.

També es va fer una recerca exhaustiva de la programació requerida per crear i enviar les dades al servidor OPC, però la dificultat de crear aquests sistemes de comunicació de zero i la falta d'un codi ja creat compatible amb els elements de què disposàvem van ajudar a prendre la decisió de sacrificar la comunicació a través d'OPC, ja que les hores dedicades no permetien avançar amb la resta d'apartats a crear i millorar.

La tasca planificada que s'hagués volgut desenvolupar queda detallada a continuació.

L'apartat de comunicació consta de dos atributs, el servidor i el client. El servidor és un proveïdor de recursos i serveis capaç d'atendre les demandes dels clients i retornar-ne respostes. El client, per altra banda, és una aplicació que es comunica i fa ús d'un servidor.

El servidor que el projecte necessita s'ha de construir amb tres paràmetres essencials que permeten la comunicació amb el nostre client, siguin aquests:

- Definició de canal, que estableix la configuració de les comunicacions, escollint així el tipus de connexió i forma de comunicació que es vol fer servir amb el client.
- Definició de dispositiu, en què es defineix quina classe de dispositiu tractarà de sol·licitar-ne serveis.
- Definició d'ítem, que consisteix en declarar les variables que es requeriran en el projecte. En el nostre cas, es voldran comunicar posició en forma de

coordenades X i Y, orientació en nombre de graus i identificador de color/forma en forma de nombre enter.

Per altra banda, el client que es vol construir es defineix amb tres parts necessàries per a la comunicació amb el servidor:

- Definició del servidor, identificant el proveïdor concret amb el que es vol establir la comunicació.
- Creació del grup, que dota d'un mecanisme que conté en forma lògica les variables amb la informació a llegir i escriure
- Definició d'ítem, que consisteix en declarar les variables que es requeriran en el projecte i que han de concordar amb les declarades al servidor.

En el nostre cas, la idea era fer servir el servidor que ofereix la controladora del robot, i construir la comunicació i crear un client amb Kepserver al mateix dispositiu que té accés a la controladora.

Un cop establerta la comunicació, l'aplicació creada escriu les variables de cada envàs detectat al servidor. El programa escrit per controlar els moviments del robot en fa la lectura, assignant els valors d'orientació i color en variables persistents capaces de ser llegides al servidor, i els valors de posició queden assignats a una variable de posició *robtarget* per poder ser utilitzada en el moviment del robot.

6.2. Resultats

Partint dels objectius inicials plantejats, avaluem quins resultats hem obtingut en el desenvolupament del projecte:

- *Crear un sistema de reconeixement d'imatge flexible que no requereixi precisió en l'ús de l'àrea de treball, ja que cal poder muntar i treure la taula de treball sovint a causa dels diversos usos del robot.*

Actualment en l'obtenció de dades a partir del processament d'imatge, la posició inicial de la taula no juga un paper important, ja que les coordenades extretes on es diposita cada envàs observat a la imatge de mostra depèn únicament de la detecció de l'envàs mateix, en cap cas de la taula. Com considerem l'àrea de treball l'espai que està a l'abast de la visió de la càmera, la taula es pot posar en qualsevol posició dins de l'àrea de treball.

- *Dissenyar un sistema de reconeixement de formes i colors amb un rang prou ampli per acceptar diferents condicions de llum del laboratori.*

Ara com ara es diferencien dues formes diferents dins de l'àrea de treball, objectes circulars i objectes amb forma de quadrilàter.
També es diferencien els colors dels envasos entre si i respecte a l'entorn, amb un percentatge d'error relativament baix quan es mantenen les condicions de llum preestablertes, que suposen tot l'enllumenat artificial del laboratori encès i les dues persianes més properes a l'àrea de treball alçades amb llum natural exterior bona o mitjana. En cas de treballar de nit s'han notat alguns errors en la distinció d'alguns envasos amb formes més irregulars a la part superior, ja que generen ombres sobre l'envàs mateix.

- *Obtenir la informació de la posició i orientació de les peces que componen la figura en coordenades del robot a partir de la imatge.*

Actualment s'obtenen les mesures de posició en variables X i Y del robot, i amb un error màxim aproximat de ±20 mm respecte a la posició original de l'envàs. L'orientació es rep en graus, i durant el posicionament l'error màxim és d'uns 10º.

- *Facilitar la comunicació entre els dos llenguatges de programació emprats.*

El procés escollit ha estat preparar les dades en el codi escrit en C# per traspassar-se amb USB i enganxar-les, sense necessitat de fer canvis, en el codi escrit en RAPID que gestiona els moviments del robot.

7. Conclusions

El projecte dut a terme permet replicar una figura en tres dimensions reconeguda mitjançant la visió artificial a partir d'implementar en una tasca robotitzada les dades obtingudes, ja que els objectius inicials han estat complerts. El sistema és capaç de reconèixer un patró reproduïble d'una figura en tres dimensions en l'espai de treball del robot, i proporciona les dades que aquest necessita per replicar la figura presentada.

Tot i que les expectatives pretenien un sistema totalment automatitzat sense la necessitat d'intervenció humana, els problemes descoberts durant el procés de creació han resultat en un procés amb un intercanvi d'informació manual. Així i tot, la solució adoptada facilita aquesta comunicació d'una forma eficient per resoldre la tasca en un temps reduït i amb una sola intervenció.

He fet un gran aprenentatge en tractament d'imatge així com en resolució de problemes tridimensionals amb informació rebuda en dues dimensions, i és un projecte amb possibilitat de ser ampliat adquirint aprenentatge en l'àmbit de la comunicació entre dispositius.

Aquest projecte m'ha permès desenvolupar un sistema similar al que es fa servir actualment a la indústria de producció automatitzada, especialment en la controlada per visió artificial.

8. Treballs futurs

8.1. Millores de precisió

La reducció de l'error de posicionament que s'obté actualment es pot fer possible afegint una conversió de dades intermèdia més complexa. Es pot fer una transformació de dades que contempli en primer lloc un espai de coordenades inclinat de tres dimensions, de manera que la visió de la càmera sigui completament perpendicular al pla XY d'aquest espai de coordenades, i seguidament fer una transformació d'aquest espai de coordenades al del robot, reduint així l'error provocat per la posició lateral de la càmera respecte a la taula de treball, i per tant, la deformació de la imatge.

8.2. Aplicació d'un OPC pel traspàs d'informació

L'aplicació d'aquest sistema sembla possible configurant un ordinador de la mateixa manera que està configurat l'ordinador del laboratori, de forma que a més de tenir instal·lat Visual Studio es pugui comunicar amb la controladora del robot i pugui fer ús del servidor OPC d'aquesta. Per dur a terme aquesta part també caldria aprendre a programar un client per realitzar el traspàs de dades amb el servidor en llenguatge C#.

8.3. Paletització amb alçada múltiple

El sistema creat només contempla fer figures en dues dimensions, ja que tot i tractar-se d'objectes tridimensionals no es fa apilament d'aquests. Un projecte possible seria permetre indicar quants elements iguals s'han d'apilar sobre cada posició. Aquest projecte, però, té dues dificultats afegides. En primer lloc, que amb una sola càmera de visió bidimensional no es poden fer càlculs espacials correctament, i per obtenir la posició del "primer pis" dels objectes apilats caldria fer la lectura en les mateixes condicions que en aquest projecte. En segon lloc, com es treballa amb envasos de diferents alçades, l'apilament només es podria realitzar amb objectes iguals.

9. Anàlisi d’impactes ambientals

Aquest projecte genera un impacte ambiental pràcticament menyspreable, ja que no es fa ús de substàncies nocives ni tecnologies perilloses per a éssers vius.

El conjunt de maquinària genera calor en el procés de dispersió d’energia sobrant, principalment en el processament de dades en la controladora del robot. El coneixement de la quantitat de calor que genera aquesta maquinària és necessari per ser contemplat a l’hora de ser instal·lat al seu lloc de treball. En el cas del laboratori on està situat, com es tracta d’un edifici climatitzat controlat per termòstats a cada sala, s’ha considerat la quantitat de calor generada en els usos de llarga duració i s’ha reduït la temperatura del termòstat amb previsió de reduir la diferència de temperatura amb l’exterior i per tant el malbaratament de recursos.

Respecte al funcionament continuat del robot, tot i que en el meu cas no ha calgut cap mena de manteniment, cal tenir present que motors fan servir olis i greixos per facilitar el lliscament entre peces. En cas de fer-se manteniment, cal tenir especial cura a l’hora de reciclar els estris de neteja, de manteniment i les restes de producte retirades fent ús d’un servei especialitzat.

10. Bibliografia

- Mario Félix Guerrero. "¿Visual Basic .NET ó C#?." *Msdn.microsoft.com*. n.d. Web. 20 Feb. 2017. https://msdn.microsoft.com/es-es/library/bb972208.aspx
- Wikipedia Contributors. "Binary large object." *Wikipedia, the Free Encyclopedia*. Wikimedia Foundation, 31 Jan. 2017. Web. 21 Feb. 2017. https://es.wikipedia.org/wiki/Binary_large_object
- N.a. "How to call function with PaintEventArgs argumnt?." *Stackoverflow.com*. n.d. Web. 30 Mar. 2017. http://stackoverflow.com/questions/11753345/how-to-call-function-with-painteventargs-argumnt
- Gonzalo Santiago. "Como convertir una imagen rgb a una imagen binaria con c#." *Simplesoftmx.blogspot.com.es*. 9 Sept. 2017. Web. 11 Sept. 2017. http://simplesoftmx.blogspot.com/2014/06/como-convertir-una-imagen-rgb-una.html
- N.a. "C# Color Table." *Flounder.com*. 1 Jul. 2013. Web. 10 Oct. 2017. http://www.flounder.com/csharp_color_table.htm#I
- N.a. "AForge.NET Framework Documentation :: Table of Content." *Aforgenet.com*. n.d. Web. 10 Nov. 2017. http://www.aforgenet.com/framework/docs/
- Codeproject. "[Solved] How to Convert 'System.Drawing.Image' to 'System.Drawing.Bitmap'? - CodeProject." *Codeproject.com*. n.d. Web. 17 Nov. 2017. https://www.codeproject.com/Questions/137139/How-to-Convert-System-Drawing-Image-to-System-Draw
- N.a. "Delete Image from PictureBox in C#." *Stackoverflow.com*. n.d. Web. 17 Nov. 2017. https://stackoverflow.com/questions/4553587/delete-image-from-picturebox-in-c-sharp
- N.a. "Cómo: Agregar o quitar referencias utilizando el cuadro de diálogo Agregar referencia." *Msdn.microsoft.com*. n.d. Web. 17 Nov. 2017. https://msdn.microsoft.com/es-es/library/wkze6zky(v=vs.120).aspx
- N.a. "Ventanas de variables locales y automáticas." *Msdn.microsoft.com*. n.d. Web. 17 Nov. 2017. https://msdn.microsoft.com/es-es/library/bhawk8xd.aspx
- Edchari. "Eror 'Font' es una referencia ambigua entre 'System.Drawing.Font' y 'iTextSharp.text.Font'." *Social.msdn.microsoft.com*. n.d. Web. 4 Dec. 2017. https://social.msdn.microsoft.com/Forums/es-ES/7e94ec1c-5ce0-41f0-96fa-0d0b29a373e3/eror-font-es-una-referencia-ambigua-entre-systemdrawingfont-y-itextsharptextfont?forum=vcses
- N.a. "AForge Blobcounter -how to count splitted objects?." *Stackoverflow.com*. n.d. Web. 6 Dec. 2017. https://stackoverflow.com/questions/29489449/aforge-blobcounter-how-to-count-splitted-objects
- Nacho Cabanes. "Programación en C# - 4.3. Estructuras o registros." *Nachocabanes.com*. n.d. Web. 8 Dec. 2017. http://www.nachocabanes.com/csharp/curso2015/csharp04c.php
- Contributors, Top. "Color Hex Color Codes." *Color-hex.com*. n.d. Web. 13 Dec. 2017. http://www.color-hex.com/
- N.a. "AForge.NET :: Detecting some simple shapes in images." *Aforgenet.com*. n.d. Web. 13 Dec. 2017. http://www.aforgenet.com/articles/shape_checker/
- Bosch APAS. "APAS inspector mobile | Bosch APAS." *Bosch APAS*. 14 Dec. 2017. Web. 27 Dec. 2017. https://www.bosch-apas.com/en/products-and-services/apas-inspector-mobile/
- N.a. "Innovaciones | Festo España." *Festo.com*. n.d. Web. 27 Dec. 2017. https://www.festo.com/cms/es_es/9457.htm
- Vision & Control Gmbh. "HMI development | Vision&Control GmbH." *Vicosys.com*. n.d. Web. 27 Dec. 2017. http://vicosys.com/en/
- N.a. "Datos del IRB 140 - IRB 140 (Robots industriales) | ABB." *New.abb.com*. 27 Dec. 2017. Web. 27 Dec. 2017. http://new.abb.com/products/robotics/es/robots-industriales/irb-140/datos

- N.a. "." *Developercenter.robotstudio.com*. 12 Oct. 2015. Web. 30 Dec. 2017. http://developercenter.robotstudio.com/BlobProxy/manuals/RapidIFDTechRefManual/doc435.html

Annex 1 Codi C#

```
1.  //*************** Llibreries pertanyents al sistema ***************//
2.  using System;
3.  using System.Collections.Generic;
4.  using System.Data;
5.  using System.Drawing;
6.  using System.Drawing.Imaging;
7.  using System.Linq;
8.  using System.Windows.Forms;
9.  using System.IO;
10. using System.Globalization;             //Es fa servir per guardar els strings
    de nombres decimals amb punt
11.
12. //*************** Llibreries pertanyents a AForge ***************//
13. using AForge;
14. using AForge.Imaging.Filters;
15. using AForge.Imaging;
16. using AForge.Math.Geometry;
17.
18. //**** Gestió de referències ambigües entre llibreries ***************//
19. using Image = System.Drawing.Image;     //Evita referència ambigua entre
    'System.Drawing.Image' i 'AForge.Imaging.Image'
20. using Point = AForge.Point;             //Evita referència ambigua entre
    'System.Point' i 'AForge.Point'
21.
22.
23. namespace ProjecteFinaldeGrau //contenidor de noms únics dins l'espai
24. {
25.
26.     public partial class ProjecteFinal : Form
27.     {
28.
29.         PictureBox quadreImatge = new PictureBox();
30.         PictureBox quadreImatge2 = new PictureBox();
31.         public Image dadaColor, imatgeInicial;
32.         public struct objecteTrobat
33.         {
34.             public double x;
35.             public double y;
36.             public double alpha;
37.             public int color;   /*  1 = Rosa
38.                                     2 = Taronja
39.                                     3 = Verd
40.                                     4 = Blau petit
41.                                     5 = Blau gran
42.                                     6 = Vermell rectangular
43.                                     7 = Vermell rodó
44.                                 */
45.         }
46.         objecteTrobat[] objecte = new objecteTrobat[10];
47.         const string
    rutaEscriptori = @"C:\Users\Ivan\Desktop\Dades.txt";       //Ruta per guardar
    dades a l'escriptori
48.         const string rutaUSB = @"F:\Dades.txt";            //Ruta per guardar
    dades a l'USB
49.         const string ruta = rutaEscriptori;
50.
51.
52.         public ProjecteFinal()
53.         {
54.
```

```
55.                 InitializeComponent();
56.                 WindowState = FormWindowState.Maximized;
57.                 this.etiqueta1.Visible = false;
58.                 this.etiqueta2.Visible = false;
59.                 this.quadreDeText.Visible = false;
60.
61.             }
62.
63.             public void clicBoto1(object sender, EventArgs e)
64.             {
65.
66.                 // Disseny de la finestra en presionar el botó del pas 1
67.
68.                 quadreImatge.Image = null;
69.                 quadreImatge2.Image = null;
70.
71.                 this.etiqueta1.Visible = true;
72.                 this.etiqueta2.Visible = true;
73.                 this.quadreDeText.Visible = true;
74.                 quadreImatge.Location = new System.Drawing.Point(690, 80);
75.                 quadreImatge2.Location = new System.Drawing.Point(30, 80);
76.
77.                 tractatIcalcul(quadreImatge);
78.                 mostraImatgeRebuda(quadreImatge2);
79.
80.                 etiqueta1.Font = new Font(etiqueta1.Font.FontFamily, 16);
81.                 etiqueta2.Font = new Font(etiqueta2.Font.FontFamily, 16);
82.
83.                 this.etiqueta1.Text = "Imatge inicial";
84.                 this.etiqueta2.Text = "Formes detectades";
85.
86.
87.
88.                 quadreDeText.Text = System.IO.File.ReadAllText(ruta);
89.
90.             }
91.
92.             public void clicBoto2(object sender, EventArgs e)
93.             {
94.
95.                 // Disseny de la finestra en presionar el botó del pas 2
96.
97.                 quadreImatge.Image = null;
98.                 quadreImatge2.Image = null;
99.
100.                this.etiqueta1.Visible = true;
101.                this.etiqueta2.Visible = true;
102.                etiqueta1.Font = new Font(etiqueta1.Font.FontFamily, 16);
103.                etiqueta2.Font = new Font(etiqueta2.Font.FontFamily, 16);
104.
105.                this.etiqueta1.Text = "Imatge inicial";
106.                this.etiqueta2.Text = "Resultat final";
107.
108.                quadreImatge2.Image = imatgeInicial;
109.                mostraImatgeRebuda(quadreImatge);
110.
111.                this.quadreDeText.Visible = false;
112.
113.            }
114.
115.            public void mostraImatgeRebuda(PictureBox quadreImatgeInici1)
116.            {
117.
```

```
118.             quadreImatgeInici1.Size = new Size(640, 480);
119.             this.Controls.Add(quadreImatgeInici1);
120.
121.             quadreImatge.Location = new System.Drawing.Point(690, 80);
122.             quadreImatge2.Location = new System.Drawing.Point(30, 80);
123.
124.             quadreImatgeInici1.Image = repImatge();      //Comunicació amb el
    port de la webcam
125.
126.         }
127.
128.
129.         public void tractatIcalcul(PictureBox quadreImatgeInici2)
130.         {
131.
132.             quadreImatgeInici2.Size = new Size(640,480);
133.             this.Controls.Add(quadreImatgeInici2);
134.
135.
136.             quadreImatgeInici2.Image = repImatge();     //Comunicació amb el
    port de la webcam
137.
138.             imatgeInicial = quadreImatgeInici2.Image;        //Guarda imatge per
    comparar al final del procés
139.
140.             quadreImatgeInici2.Image = detectaColors(); //Simplificació de la
    imatge en colors únics identificadors
141.
142.             dadaColor = quadreImatgeInici2.Image;            //Guarda imatge per
    accedir després a pixel i identificar color
143.
144.             quadreImatgeInici2.Image = filtreBN();           //Passa la imatge a
    escala de grisos i posteriorment a blanc i negre
145.
146.             quadreImatgeInici2.Image = eliminaBlobsPetits(); //Filtra els blobs
    petits i deixa els grans, els necessaris
147.
148.             quadreImatgeInici2.Image = inverteixBN();     //Inverteix els colors
    per deixar fons negre de cara a analitzar blobs
149.
150.             quadreImatgeInici2.Image = eliminaBlobsPetits(); //Filtra els blobs
    petits i deix els grans, els necessaris
151.
152.             quadreImatgeInici2.Image = eliminaSoroll();   //Suavitza el contorn
    dels blobs
153.
154.             objecte = cosProcesPrincipal();         //Dibuixat de formes
    detectades i realització de càlculs
155.
156.             guardaDades();       //Guardat de dades en un document txt
157.
158.
159.
160.         }
161.
162.         private Image repImatge()
163.         {
164.             string sourceURL = "http://10.13.16.78/record/current.jpg";
165.             byte[] buffer = new byte[100000];
166.             int llegir, total = 0;
167.
168.             // Sol·licitud d'accés al lloc web
```

```
169.             HttpWebRequest
     solicita = (HttpWebRequest)WebRequest.Create(sourceURL);
170.
171.             // Rebuda de resposta
172.             WebResponse resposta = solicita.GetResponse();
173.
174.             // Resposta stream
175.             Stream stream = resposta.GetResponseStream();
176.
177.             // Lectura de dades rebudes
178.             while ((llegir = stream.Read(buffer, total, 1000)) != 0)
179.             {
180.                 total += llegir;
181.             }
182.
183.             // Creació del mapa de bits
184.             Bitmap imatgeCam = (Bitmap)Bitmap.FromStream(new
     MemoryStream(buffer, 0, total));
185.
186.             Graphics Graf = Graphics.FromImage(imatgeCam);  //Classe graphics,
     delimita rectangle al voltant d'imatge i permet saber dades quantitatives sobre
     els píxels
187.             Graf.DrawImage(imatgeCam, 0, 0);
188.
189.             //Substitució d'adquisició de dades en cas de no estar connectat a
     la càmera, únicament per fer proves fora del laboratori
190.             //Bitmap imatgeCam = new
     Bitmap(@"C:\Users\Ivan\Dropbox\Ivan\Uni\PFG\Projecte\PFG1_0\imagenlab.jpg");
191.
192.             return imatgeCam;
193.
194.         }
195.
196.
197.
198.         private Image detectaColors()
199.         {
200.             Color pixel;
201.             Bitmap imatgeRGB = (Bitmap)quadreImatge.Image;
202.             Bitmap imatgeSimple = new Bitmap(imatgeRGB);
203.             for (int y = 0; y < imatgeRGB.Height; y++)
204.             {
205.                 for (int x = 0; x < imatgeRGB.Width; x++)
206.                 {
207.                     pixel = imatgeRGB.GetPixel(x, y);
208.
209.                     //Detecció de píxels blaus
210.
211.                     if (pixel.R > 0 & pixel.R < 155 & pixel.B > 180)
212.                         imatgeSimple.SetPixel(x, y, Color.Blue);
213.
214.                     //Detecció de píxels roses
215.
216.                     else if (pixel.R > 190 & pixel.G < 180 & pixel.B > 120) //90
217.                         imatgeSimple.SetPixel(x, y, Color.DeepPink);
218.
219.                     //Detecció de píxels vermells
220.
221.                     else
     if (pixel.R > 140 & pixel.G < 140 & pixel.B < 120)  //90
222.                         imatgeSimple.SetPixel(x, y, Color.Red);
223.
224.                     //Detecció de píxels taronges
```

```
225.
226.                     else if (pixel.R > 180 & pixel.G > 140 & pixel.B <150)
227.                         imatgeSimple.SetPixel(x, y, Color.Orange);
228.
229.                     //Detecció de píxels verds
230.
231.                     else
    if (pixel.R < 170 & pixel.G > 180 & pixel.B > 100 & pixel.B < 220)
232.                         imatgeSimple.SetPixel(x, y, Color.Green);
233.
234.
235.                     //Resta de píxels
236.                     else
237.                         imatgeSimple.SetPixel(x, y, Color.White);
238.
239.
240.                 }
241.             }
242.
243.             return imatgeSimple;
244.
245.         }
246.
247.         private Image filtreBN()
248.         {
249.
250.
251.             Grayscale grayscaleFilter = new Grayscale(0.2125, 0.7154, 0.0721);
    // Filtre Escala de grisos (BT709)
252.             Bitmap ImatgeBN = grayscaleFilter.Apply((Bitmap)quadreImatge.Image);
253.             Threshold filtreBN = new Threshold(250);       //Filtre per passar a
    blanc i negre
254.             filtreBN.ApplyInPlace(ImatgeBN);
255.
256.             return ImatgeBN;
257.
258.         }
259.
260.         private Image inverteixBN()         //Filtre inversor de blancs i negres
261.         {
262.
263.
264.             Invert filtreInversor = new Invert();
265.             Bitmap invertida = (Bitmap)quadreImatge.Image;
266.             filtreInversor.ApplyInPlace(invertida);
267.
268.             return invertida;
269.
270.         }
271.
272.
273.         private Image eliminaBlobsPetits()
274.         {
275.             //Filtratge de blobs a la imatge
276.
277.             // Creació filtre, eliminarà tots els blobs blancs menors de les
    mesures especificades a continuació, això elimina reflexos.
278.             BlobsFiltering filtreDeBlobs = new BlobsFiltering();
279.
280.             // configuració filtre
281.             filtreDeBlobs.CoupledSizeFiltering = true;
282.             filtreDeBlobs.MinWidth = 70;
283.             filtreDeBlobs.MinHeight = 50;
```

```
284.
285.             // aplicació filtre
286.             Bitmap blobs = (Bitmap)quadreImatge.Image;
287.             filtreDeBlobs.ApplyInPlace(blobs);
288.
289.             return blobs;
290.         }
291.
292.         private Image eliminaSoroll()
293.         {
294.             //Tractament imatge per eliminar soroll
295.
296.             Bitmap senseSoroll = (Bitmap)quadreImatge.Image;
297.             Median filtreMediana = new Median();
298.             filtreMediana.ApplyInPlace(senseSoroll);
299.
300.             return senseSoroll;
301.         }
302.
303.         private objecteTrobat[] cosProcesPrincipal()
304.         {
305.             objecteTrobat[] tupper = new objecteTrobat[10];
306.             var fonsQuadrat2 = imatgeInicial;
307.
308.             BitmapData bitmapData = ((Bitmap)quadreImatge.Image).LockBits(new
    Rectangle(0, 0, quadreImatge.Image.Width,
    quadreImatge.Image.Height),ImageLockMode.ReadWrite,
    quadreImatge.Image.PixelFormat);
309.
310.
311.             // Localització d'objectes
312.
313.             BlobCounter comptadorBlobs = new BlobCounter();
314.
315.             comptadorBlobs.FilterBlobs = true;
316.
317.             comptadorBlobs.ProcessImage(bitmapData);
318.             Blob[] blobs = comptadorBlobs.GetObjectsInformation();
319.             ((Bitmap)quadreImatge.Image).UnlockBits(bitmapData);
320.
321.
322.             // Comptatge d'objectes trobats i dibuixat de les formes
323.
324.             SimpleShapeChecker comprobadorForma = new SimpleShapeChecker();
325.
326.             quadreImatge.Image = new Bitmap(quadreImatge.Width,
    quadreImatge.Height);
327.             Graphics g = Graphics.FromImage(quadreImatge.Image);
328.
329.             g.DrawImage(image: fonsQuadrat2, destRect: new Rectangle(0, 0,
    quadreImatge.Width, quadreImatge.Height), srcRect: new Rectangle(0, 0,
    quadreImatge.Width, quadreImatge.Height), srcUnit: GraphicsUnit.Pixel);
330.
331.             Pen vermell = new Pen(Color.Red, 2);
332.             Pen groc = new Pen(Color.Yellow, 2);
333.
334.             int color = 0;
335.             double coordenadax = 0, coordenaday = 0;
336.             double alpha = 0, area = 0, l1 = 0, l2 = 0;
337.             double ax = 1.645, bx = 288.54, ay = 1.85173, by = -482.4136;
338.             float xsuma = 0, ysuma = 0, xbaix = 0, xdret = 0, ybaix = 0,
    ydret = 0, xesq = 0, yesq = 0, xdalt = 0, ydalt = 0;
339.             double d;
```

```
340.             Bitmap Colors;
341.
342.             for (int i = 0, n = blobs.Length; i < n; i++)
343.             {
344.                 List<IntPoint> edgePoints = comptadorBlobs.GetBlobsEdgePoints(bl
    obs[i]);
345.
346.                 int longllista = edgePoints.Count;
347.
348.                 Point centre;
349.                 float radi;
350.
351.                 if (comprobadorForma.IsCircle(edgePoints, out centre, out radi))
352.                 {
353.                     g.DrawEllipse(groc, (float)(centre.X - radi), (float)(centre
    .Y - radi), (float)(radi * 2), (float)(radi * 2));
354.
355.                     coordenadax = centre.X;
356.                     coordenadax = Math.Round(coordenadax);
357.                     coordenaday = centre.Y;
358.                     coordenaday = Math.Round(coordenaday);
359.                     alpha = 0;
360.                     color = 7;      // Vermell rodó
361.                     g.FillEllipse(Brushes.Red,(centre.X - radi), (centre.Y - rad
    i),(2 * radi),(2 * radi));
362.
363.                 }
364.                 else
365.                 {
366.
367.                     List<IntPoint> cantonades;
368.
369.                     cantonades = PointsCloud.FindQuadrilateralCorners(edgePoints
    );
370.                     g.DrawPolygon(vermell, ToPointsArray(cantonades));
371.
372.
373.                     /////////CENTRE DE QUADRILATERS////////
374.
375.                     for (int f = 0; f <= 3; f++)
376.                     {
377.                         xsuma = cantonades[f].X + xsuma;
378.                         ysuma = cantonades[f].Y + ysuma;
379.                     }
380.                     coordenadax = xsuma / 4;            // Coordenada X central
    de quadrilater
381.                     coordenadax = Math.Round(coordenadax);   // X enter
382.                     int coordx = (int)coordenadax;
383.                     coordenaday = ysuma / 4;            // Coordenada Y central
    de quadrilater
384.                     coordenaday = Math.Round(coordenaday);   // Y enter
385.                     int coordy = (int)coordenaday;
386.
387.
388.                     xbaix = cantonades[0].X;
389.                     ybaix = cantonades[0].Y;
390.                     xesq = cantonades[0].X;
391.                     yesq = cantonades[0].Y;
392.                     xdret = cantonades[0].X;
393.                     ydret = cantonades[0].Y;
394.                     xdalt = cantonades[0].X;
395.                     ydalt = cantonades[0].Y;
396.
```

```
397.                     for (int h = 0; h <= 3; h++)
398.                     {
399.                         int trobax = cantonades[h].X;
400.                         int trobay = cantonades[h].Y;
401.
402.
403.                         if (ybaix < trobay)       //trobar el pixel de més abaix
404.                         {
405.                             xbaix = trobax;
406.                             ybaix = trobay;
407.                         }
408.
409.                         if (ydalt > trobay)       //trobar el pixel de més a
    dalt
410.                         {
411.                             xdalt = trobax;
412.                             ydalt = trobay;
413.                         }
414.
415.                         if (xesq > trobax)         //trobar el pixel de més a
    l'esquerra
416.                         {
417.                             xesq = trobax;
418.                             yesq = trobay;
419.                         }
420.
421.                         if (xdret < trobax)        //trobar el pixel de més a la
    dreta
422.                         {
423.                             xdret = trobax;
424.                             ydret = trobay;
425.                         }
426.                     }
427.
428.
429.                     /////////ALPHA////////
430.
431.                     double xbe = xbaix - xesq;
432.                     double ybe = ybaix - yesq;
433.                     double xdb = xdret - xbaix;
434.                     double ybd = ybaix - ydret;
435.                     if (xbe == 0) xbe = 0.00001;
436.                     if (ybe == 0) xbe = 0.00001;
437.                     if (xdb == 0) xbe = 0.00001;
438.                     if (ybd == 0) xbe = 0.00001;
439.
440.                     d = (Math.Acos((xbe) / (Math.Sqrt(Math.Pow((xbe), 2) + Math.
    Pow((ybe), 2))))) * (180 / Math.PI);
441.                     double
    distBaixEsq = Math.Sqrt(Math.Pow((xbe), 2) + Math.Pow((ybe), 2));
442.                     double
    distDretBaix = Math.Sqrt(Math.Pow((xdb), 2) + Math.Pow((ybd), 2));
443.
444.
445. //Obtenir angle entre horitzontal de la imatge i costat llarg de l'envàs
446.
447.                     if (distBaixEsq < distDretBaix) alpha = d + 90;
448.                     else alpha = d;
449.
450.                     alpha = Math.Round(alpha);
451.
452.
453.                     /////////COLOR////////
```

```
454.
455.                     Colors = (Bitmap)dadaColor;
456.
457.                     Color pixelColor = Colors.GetPixel(coordx, coordy);
458.
459.                     if (pixelColor.R == 255 & pixelColor.G == 20 & pixelColor.B
    == 147)
460.                     {
461.                         color = 1;      // Rosa
462.                         g.FillPolygon(Brushes.DeepPink,
    ToPointsArray(cantonades));
463.                     }
464.
465.                     else
    if (pixelColor.R == 255 & pixelColor.G == 165 & pixelColor.B == 0)
466.                     {
467.                         color = 2;      // Taronja
468.                         g.FillPolygon(Brushes.Orange,
    ToPointsArray(cantonades));
469.                     }
470.
471.                     else
    if (pixelColor.R == 0 & pixelColor.G == 128 & pixelColor.B == 0)
472.                     {
473.                         color = 3;      // Verd
474.                         g.FillPolygon(Brushes.Green, ToPointsArray(cantonades));
475.                     }
476.
477.                     else
    if (pixelColor.R == 0 & pixelColor.G == 0 & pixelColor.B == 255)    // Blaus
478.                     {
479.                         l1 = (Math.Sqrt(Math.Pow((xbaix - xesq), 2) + Math.Pow((
    ybaix - yesq), 2)));
480.                         l2 = (Math.Sqrt(Math.Pow((xdret - xbaix), 2) + Math.Pow(
    (ybaix - ydret), 2)));
481.                         area = l1 * l2;
482.
483.                         if (area < 4000)
484.                         {
485.                             color = 4;      // Blau petit
486.                             g.FillPolygon(Brushes.Blue,
    ToPointsArray(cantonades));
487.                         }
488.
489.                         else
490.                         {
491.                             color = 4;      // Blau gran
492.                             g.FillPolygon(Brushes.Blue,
    ToPointsArray(cantonades));
493.                         }
494.
495.                     }
496.
497.                     else
    if (pixelColor.R == 255 & pixelColor.G == 0 & pixelColor.B == 0)
498.                     {
499.                         color = 6;      //Vermell rectangular
500.                         g.FillPolygon(Brushes.Red, ToPointsArray(cantonades));
501.                     }
502.
503.                 }
504.
505.                 double xrob = ax * coordenaday + bx;
```

```
506.                 double yrob = ay * coordenadax + by;
507.
508.                 //  Guardar a la taula les coordenades i angle
509.
510.                 tupper[i].x = xrob;
511.                 tupper[i].y = yrob;
512.                 tupper[i].alpha = alpha;
513.                 tupper[i].color = color;
514.
515.                 // Variables a zero per el següent objecte
516.
517.                 coordenadax = 0;
518.                 coordenaday = 0;
519.                 xbaix = 0;
520.                 ybaix = 0;
521.                 xdret = 0;
522.                 ydret = 0;
523.                 xesq = 0;
524.                 yesq = 0;
525.                 xsuma = 0;
526.                 ysuma = 0;
527.                 xrob = 0;
528.                 yrob = 0;
529.                 l1 = 0;
530.                 l2 = 0;
531.                 area = 0;
532.                 edgePoints = null;
533.
534.             }
535.
536.             vermell.Dispose();
537.             groc.Dispose();
538.             g.Dispose();
539.             quadreImatge.Invalidate();
540.
541.             return tupper;
542.
543.         }
544.
545.         public void guardaDades()
546.         {
547.             //Escriu les dades X, Y, alpha, color en un txt
548.
549.             if (File.Exists(ruta))
550.             {
551.                 File.WriteAllText(ruta, "");
552.             }
553.
554.             StreamWriter file = new StreamWriter(ruta, true);   //Ruta on es
    guarda l'arxiu
555.
556.
557.             file.WriteLine();       //Espai en blanc
558.             file.WriteLine();
559.
560.             //**********Guardat de les X *****************//
561.
562.             file.Write("        VAR num posx{10} := [");
563.             for (int n = 0; n < objecte.Length; n++)
564.             {
565.                 double guardar = objecte[n].x;
```

```
566.                 string guard = Convert.ToString(guardar,
      CultureInfo.InvariantCulture);      //Guardar número amb un punt "." com a
      separador de decimals
567.                 file.Write(guard);
568.                 if (n < objecte.Length - 1)
569.                 {
570.                     file.Write(", ");
571.                 }
572.             }
573.             file.Write("];");
574.             file.WriteLine();
575.
576.             //**********Guardat de les Y *****************//
577.
578.             file.Write("        VAR num posy{10} := [");
579.
580.             for (int nn = 0; nn < objecte.Length; nn++)
581.             {
582.                 double guardar = objecte[nn].y;
583.                 string guard = Convert.ToString(guardar,
      CultureInfo.InvariantCulture);      //Guardar número amb un punt com a separador
      de decimals
584.                 file.Write(guard);
585.                 if (nn < objecte.Length - 1)
586.                 {
587.                     file.Write(", ");
588.                 }
589.             }
590.             file.Write("];");
591.             file.WriteLine();
592.
593.             //**********Guardat de les alpha *****************//
594.
595.             file.Write("        VAR num posalpha{10} := [");
596.
597.             for (int m = 0; m < objecte.Length; m++)
598.             {
599.                 double guardar = objecte[m].alpha;
600.                 file.Write(guardar);
601.                 if (m < objecte.Length - 1)
602.                 {
603.                     file.Write(", ");
604.                 }
605.             }
606.             file.Write("];");
607.             file.WriteLine();
608.
609.             //**********Guardat de colors *****************//
610.
611.             file.Write("        VAR num poscolor{10} := [");
612.
613.             for (int mm = 0; mm < objecte.Length; mm++)
614.             {
615.                 double guardar = objecte[mm].color;
616.                 file.Write(guardar);
617.                 if (mm < objecte.Length - 1)
618.                 {
619.                     file.Write(", ");
620.                 }
621.             }
622.             file.Write("];");
623.             file.WriteLine();
624.
```

```
625.             file.Close();
626.
627.         }
628.
629.         private System.Drawing.Point[] ToPointsArray(List<IntPoint> points)
630.         {
631.             return points.Select(p => new System.Drawing.Point(p.X,
    p.Y)).ToArray();
632.         }
633.
634.     }
635.
636. }
```

Annex 2 Codi RAPID

```
1.  MODULE MainModule
2.
3.
4.          CONST robtarget rosa:=[[114.69,420.67,127.08],[1.14218E-05,0.707104,0.707109,-
                  1.7807E-05],[0,-1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
5.          CONST robtarget srosa:=[[114.69,420.67,327.08],[1.14218E-05,0.707104,0.707109,-
                  1.7807E-05],[0,-1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
6.          CONST robtarget taronja:=[[-92.79,423.56,162.23],[3.16312E-05,0.707095,0.707119,-
                  2.8133E-05],[1,0,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
7.          CONST robtarget staronja:=[[-92.79,423.56,362.23],[3.16312E-05,0.707095,0.707119,-
                  2.8133E-05],[1,0,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
8.          CONST robtarget verd:=[[-250.77,419.70,128.89],[5.47909E-06,-0.0280297,0.999607,-
                  2.00553E-05],[1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
9.          CONST robtarget sverd:=[[-250.77,419.70,328.89],[5.47909E-06,-0.0280297,0.999607,-
                  2.00553E-05],[1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
10.         CONST robtarget blaup:=[[-380.46,394.63,152.20],[2.80487E-05,0.707118,0.707096,-
                  4.21468E-06],[1,0,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
11.         CONST robtarget sblaup:=[[-380.46,394.63,352.20],[2.80487E-05,0.707118,0.707096,-
                  4.21468E-06],[1,0,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
12.         CONST robtarget blaug:=[[98.08,580.33,140.86],[2.9861E-05,0.707117,0.707096,-
                  8.51366E-06],[0,-1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
13.         CONST robtarget sblaug:=[[98.08,580.33,340.86],[2.9861E-05,0.707117,0.707096,-
                  8.51366E-06],[0,-1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
14.         CONST robtarget vermell:=[[-128.69,576.82,122.20],[5.11241E-05,0.707109,0.707105,-
                  2.51827E-05],[1,-1,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
15.         CONST robtarget svermell:=[[-128.69,576.82,322.20],[5.11241E-05,0.707109,0.707105,-
                  2.51827E-05],[1,-1,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
16.         CONST robtarget rodo:=[[-341.75,560.18,128.60],[5.51281E-05,0.707107,0.707106,-
                  2.29068E-05],[1,0,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
17.         CONST robtarget srodo:=[[-341.75,560.18,328.60],[5.51281E-05,0.707107,0.707106,-
                  2.29068E-05],[1,0,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
18.
19.         CONST robtarget home:=[[514.10,-
                  1.71,709.46],[0.681848,0.00591353,0.731464,0.0030479],[-1,-
                  1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
20.         CONST robtarget repos:=[[-92.79,490,600],[3.16312E-05,0.707095,0.707119,-2.8133E-
                  05],[1,0,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
21.
22.         VAR robtarget posicio;
23.         VAR robtarget sposicio;
24.
25.         !VAR Agafa := AO;              ! Representació d'una sortida analògica que activa la
                  succió de l'eina
26.
27.         VAR num x;
28.         VAR num y;
29.         VAR num z;
30.         VAR num alpha;
31.         VAR num color;
32.
33. !****************ENTRADA DE DADES DE CADA PEÇA**********************
34.
35.
36.         VAR num posx{10} := [610,500.355,400,550,0,0,0,0,0,0];
37.         VAR num posy{10} := [400,200,-40.33,300,0,0,0,0,0,0];
38.         VAR num posalpha{10} := [80,40,110,0,0,0,0,0,0,0];
39.         VAR num poscolor{10} := [1,6,4,7,0,0,0,0,0,0];
40.
41.
42.
```

```
!*****************INICI DE PROGRAMA***********************************


        PROC main()

        MOVEJ repos,v1000,fine,tool0;                   !Posició a la espera de recollida


        FOR piece FROM 1 to 10 DO

                x := posx{piece};
                y := posy{piece};
                alpha := posalpha{piece};
                color := poscolor{piece};


!*****************POSICIONAMENT DE PEÇA*****************************

                IF color = 1 THEN               !Envàs ROSA

                        z := 260.8;
                        posicio :=[[x,y,z],[0,0,1,0],[0,-
                1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
                        sposicio :=[[x,y,z+200],[0,0,1,0],[0,-
                1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

                        MOVEJ srosa,v1000,fine,tool0;
                        MOVEL rosa,v100,fine,tool0;
                        !Set Agafa;
                        WaitTime 1;
                        MOVEL srosa,v100,fine,tool0;
                        MOVEJ RelTool (sposicio, 0,0,0\Rz:=alpha), v1000, fine, tool0;
                        MOVEJ
                RelTool (posicio, 0,0,0\Rz:=alpha), v100, fine, tool0;         !Gira els graus
                marcats per alpha
                        !Reset Agafa;
                        WaitTime 1;

                        MOVEJ repos,v1000,fine,tool0;
                        WaitTime 1;


                 ELSEIF color = 2 THEN          !Envàs TARONJA

                        z := 297.19;
                        posicio :=[[x,y,z],[0,0,1,0],[0,-
                1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
                        sposicio :=[[x,y,z+200],[0,0,1,0],[0,-
                1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

                        MOVEJ staronja,v1000,fine,tool0;
                        MOVEL taronja,v100,fine,tool0;
                        !Set Agafa;
                        WaitTime 1;
                        MOVEL staronja,v100,fine,tool0;
                        MOVEJ RelTool (sposicio, 0,0,0\Rz:=alpha), v1000, fine, tool0;
                        MOVEJ
                RelTool (posicio, 0,0,0\Rz:=alpha), v100, fine, tool0;         !Gira els graus
                marcats per alpha
                        !Reset Agafa;
                        WaitTime 1;

```

```
98.                         MOVEJ repos,v1000,fine,tool0;
99.                         WaitTime 1;
100.
101.
102.                 ELSEIF color = 3 THEN           !Envàs VERD
103.
104.                         z := 265.04;
105.                         posicio :=[[x,y,z],[0,0,1,0],[0,-
                    1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
106.                         sposicio :=[[x,y,z+200],[0,0,1,0],[0,-
                    1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
107.
108.                         MOVEJ sverd,v1000,fine,tool0;
109.                         MOVEL verd,v100,fine,tool0;
110.                         !Set Agafa;
111.                         WaitTime 1;
112.                         MOVEL sverd,v100,fine,tool0;
113.                         MOVEJ RelTool (sposicio, 0,0,0\Rz:=alpha), v1000, fine, tool0;
114.                         MOVEJ
                    RelTool (posicio, 0,0,0\Rz:=alpha), v100, fine, tool0;         !Gira els graus
                    marcats per alpha
115.                         !Reset Agafa;
116.                         WaitTime 1;
117.
118.                         MOVEJ repos,v1000,fine,tool0;
119.                         WaitTime 1;
120.
121.
122.                 ELSEIF color = 4 THEN           !Envàs BLAU PETIT
123.
124.                         z := 287.35;
125.                         posicio :=[[x,y,z],[0,0,1,0],[0,-
                    1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
126.                         sposicio :=[[x,y,z+200],[0,0,1,0],[0,-
                    1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
127.
128.                         MOVEJ sblaup,v1000,fine,tool0;
129.                         MOVEL blaup,v100,fine,tool0;
130.                         !Set Agafa;
131.                         WaitTime 1;
132.                         MOVEL sblaup,v100,fine,tool0;
133.                         MOVEJ RelTool (sposicio, 0,0,0\Rz:=alpha), v1000, fine, tool0;
134.                         MOVEJ
                    RelTool (posicio, 0,0,0\Rz:=alpha), v100, fine, tool0;         !Gira els graus
                    marcats per alpha
135.                         !Reset Agafa;
136.                         WaitTime 1;
137.
138.                         MOVEJ repos,v1000,fine,tool0;
139.                         WaitTime 1;
140.
141.
142.                 ELSEIF color = 5 THEN           !Envàs BLAU GRAN
143.
144.                         z := 274.6;
145.                         posicio :=[[x,y,z],[0,0,1,0],[0,-
                    1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
146.                         sposicio :=[[x,y,z+200],[0,0,1,0],[0,-
                    1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
147.
148.                         MOVEJ sblaug,v1000,fine,tool0;
149.                         MOVEL blaug,v100,fine,tool0;
150.                         !Set Agafa;
```

```
151.                    WaitTime 1;
152.                    MOVEL sblaug,v100,fine,tool0;
153.                    MOVEJ RelTool (sposicio, 0,0,0\Rz:=alpha), v1000, fine, tool0;
154.                    MOVEJ
                RelTool (posicio, 0,0,0\Rz:=alpha), v100, fine, tool0;         !Gira els graus
                marcats per alpha
155.                    !Reset Agafa;
156.                    WaitTime 1;
157.
158.                    MOVEJ repos,v1000,fine,tool0;
159.                    WaitTime 1;
160.
161.
162.              ELSEIF color = 6 THEN           !Envàs VERMELL
163.
164.                    z := 257.74;
165.                    posicio :=[[x,y,z],[0,0,1,0],[0,-
                1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
166.                    sposicio :=[[x,y,z+200],[0,0,1,0],[0,-
                1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
167.
168.                    MOVEJ svermell,v1000,fine,tool0;
169.                    MOVEL vermell,v100,fine,tool0;
170.                    !Set Agafa;
171.                    WaitTime 1;
172.                    MOVEL svermell,v100,fine,tool0;
173.                    MOVEJ RelTool (sposicio, 0,0,0\Rz:=alpha), v1000, fine, tool0;
174.                    MOVEJ
                RelTool (posicio, 0,0,0\Rz:=alpha), v100, fine, tool0;         !Gira els graus
                marcats per alpha
175.                    !Reset Agafa;
176.                    WaitTime 1;
177.
178.                    MOVEJ repos,v1000,fine,tool0;
179.                    WaitTime 1;
180.
181.
182.              ELSEIF color = 7 THEN           !Envàs RODÓ
183.
184.                    z := 263.05;
185.                    posicio :=[[x,y,z],[0,0,1,0],[0,-
                1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
186.                    sposicio :=[[x,y,z+200],[0,0,1,0],[0,-
                1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
187.
188.                    MOVEJ srodo,v1000,fine,tool0;
189.                    MOVEL rodo,v100,fine,tool0;
190.                    !Set Agafa;
191.                    WaitTime 1;
192.                    MOVEL srodo,v100,fine,tool0;
193.                    MOVEJ sposicio,v1000,fine,tool0;
194.                    MOVEJ posicio,v100,fine,tool0;
195.                    !Reset Agafa;
196.                    WaitTime 1;
197.
198.                    MOVEJ repos,v1000,fine,tool0;
199.                    WaitTime 1;
200.
201.
202.              ELSE
203.                    !Res
204.
205.
```

```
206.                 ENDIF
207.         ENDFOR
208.
209.         ENDPROC
210. ENDMODULE
```